UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Agentes virtuales con capacidades cognitivas utilizando IBM Watson

Manuel Esteban Carrillo Calderón
Tutor: Jenaro Gallego Gómez
Ponente: Estrella Pulido Cañabate

Junio 2017

Agentes virtuales con capacidades cognitivas utilizando IBM Watson

AUTOR: Manuel Esteban Carrillo Calderón
TUTOR: Jenaro Gallego Gómez
PONENTE: Estrella Pulido Cañabate

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2017

Resumen (castellano)

En la actualidad, los avances en la tecnología informática y la creciente globalización por medio de Internet y las redes sociales han obligado a que los comercios tradicionales luchen por digitalizarse, a la par que los comercios online traten de ser cada vez más personales y cercanos a los clientes. En esto consiste el **comercio electrónico conversacional,** una evolución del ecosistema del comercio electrónico.

Hoy en día, los chats automáticos con mensajes estándar no cumplen con los requisitos de unos clientes cada vez más exigentes con sus necesidades a la hora de solicitar alguna clase de servicio (recomendación de productos, atención personalizada y en tiempo real, etc.), y la atención humano-a-humano presenta limitaciones económicas y temporales poco rentables para un negocio. Es aquí donde interviene la principal herramienta que hace posible el comercio electrónico conversacional: los **agentes virtuales con capacidades cognitivas**.

Este Trabajo de Fin de Grado tiene como objetivo principal el diseño y desarrollo de un sistema conversacional, **Watson AV**, capaz de analizar e interpretar diversos tipos de datos, incluyendo texto no estructurado y audio para interactuar con una persona y proporcionar recomendaciones individualizadas mediante la comprensión de su personalidad, su tono y emoción.

El entorno de desarrollo de esta aplicación se basa en los servicios de Watson Developer Cloud alojados en la plataforma IBM Bluemix y ofrecidos a través de una API Rest. La tecnología utilizada incluye, inicialmente, IBM Watson Conversation y los servicios de Speech to Text y Text to Speech. Watson Conversation incluye su propio entorno de desarrollo a través del cual se construye la conversación indicando las intenciones (acciones) a identificar en el diálogo y las principales entidades (objetos o valores) con las que eliminar la ambigüedad en la conversación. Todos estos servicios se integran a través de Node-Red, un entorno de diseño gráfico de flujos de integración basado en node.js que permite desplegar a la aplicación en un servidor web y mantenerla accesible a través de un navegador.

Aunque el asistente virtual Watson AV construido en este proyecto está enfocado a un área específica (panel de control de un automóvil), se puede ver como una base adaptativa para la construcción de agentes virtuales dedicados a diversas áreas de la sociedad.

Palabras clave (castellano)

Agente/Asistente Virtual, Comercio electrónico, Capacidad cognitiva, Interacción hombre-máquina, Bot conversacional.

Abstract (English)

Nowadays, progress in information technology and the worldwide growth through Internet and social networks have forced traditional businesses to struggle to digitize, while online shops try to be more personal and close to the clients. This is the **conversational electronic commerce**, an evolution of the e-commerce ecosystem.

Today, automatic chats with standard messages do not get the requirements of increasingly demanding customers with their needs when they request some kind of service (product recommendation, personalized and real-time attention, etc.), and human-to-human care has economic limits that are unprofitable for a business. Here the main tool that makes conversational electronic commerce possible intervenes: **virtual agents with cognitive abilities**.

This bachelor thesis has as main objective the design and development of a conversational system, **Watson AV**, able to analyse and interpret diverse types of data, including unstructured text and audio to interact with people and to provide individualized recommendations through the understanding of their personality, tone and emotion.

The development environment for this application is based on Watson Developer Cloud services hosted on the IBM Bluemix platform and offered through a Rest API. The technology used initially includes IBM Watson Conversation and the services of Speech to Text and Text to Speech. Watson Conversation includes its own development environment through which the conversation is built by indicating the intentions (actions) to be identified in the dialogue and the main entities (objects or values) with which the ambiguity in the conversation can be eliminated. These services are integrated through Node-Red, a node.js-based integration flow graphic design environment that allows you to deploy the application into a web server and keep it accessible through a browser.

Although the virtual assistant Watson AV built on this project is focused on a specific area (control panel of a car), it can be seen as an adaptive base for the construction of virtual agents dedicated to various areas of society.

Keywords (inglés)

Virtual Agent/Assistant, Electronic commerce (e-commerce), Cognitive ability, Human-machine interaction, Chatbot.

Agradecimientos

En primer lugar, agradecer a mi familia por todo el apoyo que me ha brindado durante todos los años de formación profesional. Gracias por dejarme elegir mi camino, por ayudarme en todo lo posible y por estar siempre a mi lado.

A mis tutores Jenaro Gallego y Estrella Pulido por su paciencia, apoyo y dedicación a lo largo de todo el período de tiempo de realización del proyecto, otorgándome los recursos necesarios sin los cuáles no habría sido posible sacar este trabajo adelante.

Gracias a mis amigos de dentro y fuera de la universidad por su apoyo y por todas las experiencias y momentos compartidos que me han ayudado a crecer como persona.

Gracias también a todas las personas que evaluaron y juzgaron el trabajo realizado, permitiéndome corregir y aprender de mis errores.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

El desarrollo de agentes virtuales está siendo una de las aplicaciones más habituales de las tecnologías cognitivas. Es un ámbito en el que se pone en práctica los avances en la comprensión del lenguaje natural y las capacidades de interacción automáticas.

En los últimos años, el número de tecnologías cognitivas dedicadas al diseño y desarrollo de bots conversacionales ha aumentado considerablemente gracias, en gran medida, al crecimiento de la compra y venta de productos o servicios a través de medios electrónicos como Internet u otras redes informáticas (el denominado comercio electrónico o *e-commerce*) donde el contacto de los usuarios con las empresas es constante y atender a los clientes de manera inmediata en cualquier momento es un requisito indispensable para mejorar la calidad del servicio.

"Las instituciones que puedan implementar una estrategia de soporte a usuarios en múltiples niveles, podrán limitar el incremento de los costos de servicio (que en un 60% están asociados a personal), aumentar el número de contactos y conservar la lealtad del cliente"[1]

La aplicación que se presenta, **Watson AV,** es un sistema de conversación que procesa la información que proporciona el usuario (preguntas o comentarios) elaborando una respuesta adecuada. Todo ello basado en técnicas de procesamiento de lenguaje natural e inteligencia artificial.

En el caso de este proyecto, la aplicación sirve de ayuda para ejecutar las múltiples tareas que dispone el panel de control de un automóvil, aunque la manera en la que se diseña brinda flexibilidad para que la aplicación sea capaz de adecuarse y ofrecer cualquier otro tipo de servicio.

1.2 Objetivos

El objetivo de este trabajo de fin de grado es, por tanto, diseñar e implementar una aplicación capaz de dialogar con usuarios y resolver sus dudas y cuestiones de forma natural, de manera escrita o mediante voz, y en tiempo real, utilizando los servicios de IBM Watson.

Todos estos objetivos se concentran en **Watson AV,** el agente virtual desarrollado para este proyecto para ayudar en la recomendación de servicios con el que se podrá interactuar por voz o por teclado a través del navegador, siguiendo el siguiente esquema de desarrollo:

- Utilización de la plataforma de desarrollo Bluemix.
- Configuración de servicios cognitivos a través de IBM Watson Conversations.
- Inclusión de servicios de voz a texto y viceversa: Speech to Text – Text to Speech.
- Integración de todos los servicios a través de la herramienta Node-Red.
- Uso de todo el servicio a través de navegador.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1:** Introducción (actual). Se presenta la motivación y objetivos del proyecto que se va a desarrollar.
- **Capítulo 2:** Estado del Arte. En este capítulo se define el concepto de agente virtual, se detalla su situación actual, su origen y las aplicaciones de este tipo que existen en la actualidad. También se presenta la tecnología utilizada en este proyecto para construir un agente virtual.
- **Capítulo 3:** Diseño. En este capítulo se abarca el diseño de la aplicación, mostrando los módulos en que se divide, cómo se realiza la integración e interacción entre ellos y también el plan de desarrollo seguido.
- **Capítulo 4:** Desarrollo. En esta sección se explica cómo se desarrollan e implementan los módulos presentes en la aplicación, adentrándose en aspectos técnicos de programación, así como en la forma en la que crean y utilizan los servicios cognitivos.
- **Capítulo 5:** Integración, pruebas y resultados. Aquí se muestran las pruebas realizadas individualmente sobre los módulos desarrollados, así como las pruebas de integración en entorno real de la aplicación.
- **Capítulo 6:** Conclusiones y trabajo futuro. Como punto final, se detallan las conclusiones obtenidas del proyecto, además de un breve planteamiento sobre las líneas de trabajo futuro.

2 Estado del arte

El auge del comercio electrónico, las redes sociales y el llamado internet de las cosas (interconexión digital de objetos cotidianos con internet) obligan a buscar nuevas formas de satisfacer los requisitos de unos clientes cada vez más exigentes con sus necesidades a la hora de solicitar alguna clase de servicio.

Según un estudio realizado por OBS Bussines School, en el año 2020 más de 30 mil millones de dispositivos estarán conectados a Internet [2] por lo que la manera de consumir será, cada vez más, a través de chatbots. Es aquí donde entra la inteligencia artificial y el desarrollo de sistemas conversacionales (agentes virtuales) que sean capaces de "entender" los deseos y necesidades de los usuarios y responder adecuadamente a ellos.

"En 2020 tendremos más conversaciones con bots que con nuestros novios"
Rebeca G. Marciel [3]

En esta sección se va a hacer una introducción al concepto de *Agente Virtual*, resaltando las características que posee y las áreas en las que se emplea. Posteriormente, se hará un recorrido sobre algunos de los asistentes virtuales y chatbots desarrollados a través del tiempo y también sobre los que actualmente se encuentran en el mercado, finalizando con la presentación de la tecnología cognitiva utilizada en este proyecto.

2.1 Agente Virtual

2.1.1 Definición

Un Agente Virtual (AV) es un sistema capaz de interactuar de manera natural con una persona para resolver sus dudas, ofrecerle algún tipo de servicio o realizar una acción que ella solicite. Ejemplos de Agentes Virtuales pueden ser los sistemas de mensajería y comunicación, como puede ser un sitio web que ofrece servicios, una red social, un *call center*, un robot, etc.

La evolución de este tipo de sistemas ha transcurrido paralelamente al auge del comercio electrónico o *e-commerce*, (un método de compraventa de bienes, productos y servicios valiéndose de internet como medio [4]) y nace de la exigencia por parte de los clientes de formas de comunicación inmediatas y de la capacidad de mantenerse interconectados en todo momento. Muchas de las compañías dedicadas a este tipo de comercio ven en los agentes virtuales una forma de salvar las limitaciones que presentan los buscadores convencionales implantados en sus sitios web, permitiendo al sistema comunicarse de forma natural con los usuarios, es decir, manteniendo un estilo de interacción eminentemente humano (más si la conversación es mediante voz), contextualizando e interpretando de una manera precisa las solicitudes que éstos emiten y cumpliendo con la demanda de forma eficaz, sin perder las ventajas del canal digital: mucho menor coste, actividad 24x7, eliminando las limitaciones de horario, operativa instantánea sin esperas, etc.

2.1.2 Ventajas

Las principales ventajas que presenta el uso de agentes virtuales en una empresa o sistema que ofrece un servicio son:

1. **Omnicanalidad:** Poseen la capacidad de interactuar con los usuarios adaptando su comunicación al canal que se esté utilizando, como puede ser voz, texto o incluso mediante imágenes, mejorando la satisfacción general del cliente al permitir comunicación diversa.
2. **Disponibilidad:** Los asistentes virtuales se encuentran desplegados a través de la red, por lo que son accesibles para todo el mundo y en cualquier momento (24/24 horas - 7/7 días), evitando las restricciones laborales y horarias, y cubriendo con esto la necesidad de interconexión total que en la era actual los usuarios reclaman.
3. **Capacidad de aprendizaje:** Poseen aprendizaje progresivo a partir de las consultas de los usuarios y de la interacción que mantienen con ellos, incrementando con el tiempo la calidad de sus respuestas. Esta capacidad de entendimiento, unido al **machine learning** (rama dentro del campo de la Inteligencia Artificial cuyo objetivo es dotar a los ordenadores de capacidad de aprendizaje automático) aumenta la precisión y agiliza el proceso de recomendación de servicios.
4. **Objetividad:** Los AV`s dialogan de un modo natural y objetivo con el cliente para descubrir cuál es su verdadera necesidad. Las conversaciones tienen como propósito interpretar la intención del usuario para resolver su necesidad puntual.
5. **Eficacia en las consultas:** Las capacidades cognitivas de los agentes virtuales les permiten realizar un análisis sintáctico, morfológico y semántico de cualquier petición, frase o palabra introducida por el usuario. Esta funcionalidad aporta una referencia más precisa para el usuario en contraposición a los buscadores tradicionales.
6. **Personalización:** La captura de toda la información (datos) de comportamiento de la interacción entre el asistente virtual y los usuarios, unido al procesamiento del lenguaje natural y los algoritmos de aprendizaje automático consiguen dar una respuesta de alta fidelidad a las preguntas y peticiones realizadas por los clientes. Según Hardvard Bussines Review, la personalización en la recomendación de un servicio a los clientes mejora la facturación entre un 5 y un 15% y mejora la eficacia del gasto en acciones de marketing entre un 10 y un 30%; de hecho, el 44% de los consumidores de EE.UU prefieren bots conversacionales en lugar de humanos para las relaciones comerciales [5].

La Figura 2-1 muestra algunos de los canales mediante los que se puede comunicar un asistente virtual.

Figura 2-1: Canales de comunicación de un AV

2.1.3 Casos de uso

Como se acaba de ver, el uso de agentes virtuales proporciona múltiples ventajas a empresas o sistemas que ofrecen un servicio. Los casos de uso más comunes actualmente son:

- **Contact center:** Se utiliza para facilitar a los operadores del centro de contacto la herramienta ideal para ofrecer una respuesta única de manera automática.
- **Web de empresa:** Utilizados para proporcionar una respuesta rápida y precisa a las preguntas de sus clientes desde su sitio Web.
- **Redes sociales:** El uso de asistentes virtuales en redes sociales de una compañía asegura que la respuesta es la información oficial, equivalente a la ofrecida en otros canales de comunicación de la misma empresa.
- **Vehículos:** El uso de agentes virtuales en vehículos a motor como automóviles, autobuses o aviones, facilita a los conductores controlar el vehículo únicamente dictando órdenes al AV, evitando las distracciones y mejorando la conducción.
- **Dispositivos móviles:** La actual proliferación de aplicaciones para dispositivos móviles y su uso motiva la utilización de AV's que faciliten a los clientes hacer preguntas en cualquier dispositivo y recibir una respuesta simple y precisa.

2.2 Agentes Virtuales existentes

2.2.1 Bots conversacionales clásicos

El concepto de asistente virtual es relativamente nuevo, pero desde hace mucho tiempo se ha venido trabajando en sistemas informáticos capaces de mantener una conversación con un ser humano, intentando, en un principio, superar con éxito el famoso *Test de Turing*, que prueba la habilidad de la máquina de exhibir un comportamiento inteligente similar, o indistinguible, del de un humano [6].

Estos primeros sistemas reciben el nombre de *bots conversacionales clásicos*. Entre los bots más clásicos y tempranos se encuentran: Eliza, SmarterChild, Parry, SHRDLU, Racter, A.L.I.C.E o Jabberwacky [7].

A continuación, se comentan las principales características de algunos de ellos.

2.2.1.1 Eliza (1966)

Eliza fue uno de los primeros sistemas informáticos que contaba con la capacidad de procesar el lenguaje natural humano para llevar a cabo una conversación mediante texto de forma coherente con una persona.

Funcionaba de forma básica buscando palabras clave dentro de una frase u oración escrita por el usuario del sistema. Si reconocía alguna palabra en su base de datos, contestaba con una o varias respuestas pre-programadas para esa entrada. Si no reconocía la palabra, Eliza utilizaba varios tucos para hacer creer a sus interlocutores de que seguía el tema de la conversación, por ejemplo, contestaba con ambigüedades como "Mmmm, qué interesante", "Cuéntame más". En ocasiones devolvía la misma entrada del usuario intercambiando pronombres personales como "tú" por "yo", envolviendo el texto del usuario en una frase más amplia, o reformulando la entrada en forma de pregunta.

Este programa fue diseñado y desarrollado en el Instituto Técnico de Massachusetts (MIT) por Joseph Weizenbaum entre 1964 y 1966. De manera general, este sistema se puede construir fácilmente haciendo uso de dos bases de datos: una para almacenar las entradas del usuario y otra para las respuestas asociadas. Debido a la baja complejidad que presenta, este programa ha sido implementado en una gran variedad de lenguajes de programación.

La Figura 2-2 muestra un ejemplo de una conversación típica entre un ser humano y este programa. En ella se pueden apreciar los patrones de respuesta de Eliza antes comentados.

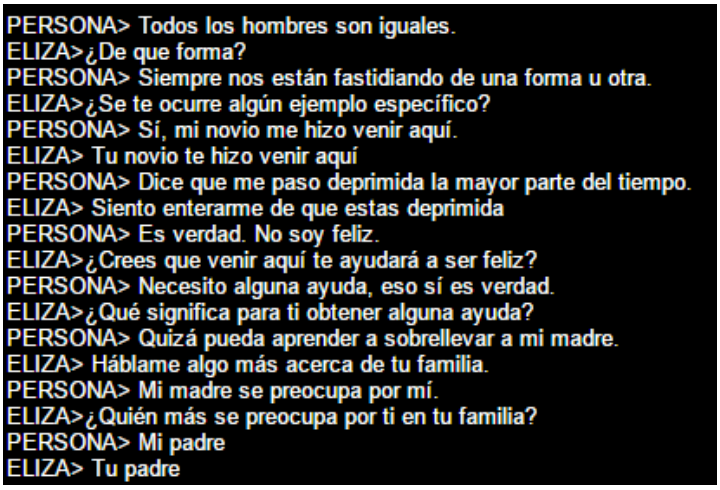


Figura 2-2: Ejemplo de conversación con Eliza

2.2.1.2 Jabberwacky (1981)

Este bot nació en 1981 como un proyecto que intentaba crear una inteligencia artificial a partir de la interacción humana. Se fue desarrollando con el tiempo incluyendo varias actualizaciones y mejoras que le llevaron a ganar los más altos lugares de la competición anual de inteligencia artificial **Loebner Prize**.

Estaba diseñado para aprender a comunicarse en varios idiomas a través de su interacción con personas, almacenaba las entradas y respuestas proporcionadas por los usuarios en la conversación para construir a partir de ellas respuestas que se adecuaran a la situación y contexto de futuras conversaciones.

La Tabla 2-1 muestra la cronología de este asistente virtual.

Fecha	Acontecimiento
1981	Nacimiento del proyecto. Desarrollado por Rollo Carpenter
1997	Lanzamiento en Internet con el nombre Jabberwacky
Oct. 2003	Obtiene el tercer lugar en Loebner Prize
Sep. 2004	Obtiene el segundo lugar en Loebner Prize
Sep. 2005	George, un personaje de Jabberwacky, gana el premio Loebner Prize
Sep. 2006	Vuelve a ganar el premio Loebner

Oct. 2008	Una nueva versión del programa se pone en marcha con el nombre de **CleverBot**

Tabla 2-1: Cronología de Jabberwacky [8]

2.2.1.3 A.L.I.C.E (1995)

Este bot nace como parte del llamado *Proyecto Pandora*, un proyecto de internet que consiste en la creación de chatbots especializados en diferentes temáticas. A.L.I.C.E (Artificial Linguistic Internet Computer Entity) simula una conversación inteligente y desarrollada capaz de engañar a un usuario haciéndole pensar que está hablando con otro ser humano.

Fue diseñada en el lenguaje de programación Java por el Dr. Richard S. Wallace, y usa un lenguaje bsado en XML de nombre AIML (Artificial Intelligence Markup Language) para especificar las reglas heurísticas de la conversación. Está inspirado en Eliza, y ha conseguido ganar la mencionada competición Loebner Prize en el año 2000, 2001 y 2004 [9].

2.2.2 Actualidad de los AV's

En los últimos años se ha experimentado un *boom* en lo que a desarrollo de bots conversacionales se refiere, impulsados por el avance de la inteligencia artificial (especialmente en el procesamiento del lenguaje natural), por las redes sociales y nuevos hábitos de los consumidores. Todos estos avances y cambios han transformado la manera en la que se veían estos sistemas (programas de diálogo para entretener) y han pasado de ser simples *chatbots* a convertirse en los *Agentes Virtuales* comentados al inicio de esta sección.

En la actualidad hay una gran cantidad de estos sistemas aplicados y especializados en distintos campos. A continuación, se van a comentar algunos de los AV's más destacados según el área en el que se desempeñan.

2.2.2.1 AV para asistencia humana en Web

Las compañías que prestan algún tipo de servicio online utilizan asistentes virtuales sobre todo en el ámbito de **atención al cliente**, aunque en ocasiones se utilizan como secretarios virtuales para ofrecer servicios internos a sus empleados. Algunos ejemplos son:

- **Renfe:** La página web de Renfe (principal operadora de red ferroviaria de Espana) cuenta con un asistente virtual llamado **Irene** para ayudar a los usuarios con cuestiones sobre trayectos y horarios.
- **Iberia:** Esta aerolínea española también cuenta con su propio asistente virtual en su sitio web para ayudar a los usuarios con sus consultas sobre vuelos, precios, equipaje, servicios abordo, etc.
- **Correos:** La compañía de correos, al igual que las anteriores, cuenta con un asistente virtual de nombre **Sara** que ayuda a los clientes en tareas como localizar envíos, localizar oficinas cercanas o ayuda en el envío de paquetes.

- **El Mundo:** Aunque ahora mismo no se encuentra disponible, este periódico incorporó en su página web un chatbot para ayudar en la consulta de los premios obtenidos en la Lotería de Navidad.

2.2.2.2 AV en dispositivos móviles

El avanzado desarrollo de la tecnología aplicada a dispositivos móviles permite la incorporación de asistentes virtuales en ellos que ayuden a los usuarios a realizar determinadas tareas en el dispositivo. Los ejemplos más destacados en este campo son:

- **Siri:** Es una aplicación creada por Apple que tiene la función de asistente virtual en dispositivos iOS, como iPhone, iPad, Mac, Apple Watch y Apple TV, integrados por defecto. Utiliza el lenguaje natural para responder preguntas de los usuarios, hacer recomendaciones, acceder y realizar cambios sobre los dispositivos y realizar acciones mediante la delegación de solicitudes hacia un conjunto de servicios web.
- **Cortana:** Es el asistente personal inteligente desarrollado por Microsoft disponible para las plataformas Windows Phone 8.1, Windows 10, Windows 10 Mobile, Microsoft Band, Microsoft Band 2, CyanogenMod, Xbox One, integrador por defecto; y para Android e iOS disponibles para su descarga. Es capaz de reconocer el lenguaje natural y utiliza Bing, Yelp y Foursquare como motor de búsqueda y bases de datos.
- **Google Now:** Es el agente virtual desarrollado por la compañía Google. Se encuentra disponible para dispositivos Android dentro de la aplicación de Google Search. Como los AV's comentados, utiliza una interfaz de lenguaje natural para responder preguntas, hacer recomendaciones y realizar acciones en el dispositivo. Ofrece al usuario información que puede querer, tomando en cuenta sus hábitos de búsqueda.

La Tabla 2-2 muestra una comparativa de las funciones que pueden realizar los asistentes virtuales comentado en un dispositivo móvil.

Función	Siri	Cortana	Google Now
Abrir aplicaciones	Si	Si	Si
Previsión del tiempo	Si	Si	Si
Calendario	Si	Si	Si
Configuración de alarmas	Si	Si	Si
Escribir consultas	Si	No	Si
Recordatorios	Si	No	No
Acceder a las funciones de las aplicaciones	Si	No	No
Llamadas	Si	Si	Si
Envío de mensajes o emails	Si	Si	Si
Reproducción de música	Si	Si	Si
Reconocimiento de música	Si	No	Si

Tecnología de búsqueda	Bing	Bing, Wolfram Alpha	Google
Sentido del humor	Si	Si	No

Tabla 2-2: Comparativa de AV's móviles

2.2.2.3 AV en redes de mensajería

En la actualidad las personas dedican mucho tiempo a interactuar con sus contactos a través de redes sociales y de mensajería. Debido a esto, varias empresas han incluido bots conversacionales dentro de sus cuentas en las redes sociales para prestar atención al cliente. Algunos ejemplos son:

- **Destinia:** Esta agencia de viajes online cuenta con un AV dentro del servicio de mensajería Facebook Messenger que permite a sus clientes buscar hoteles dentro de la plataforma, aunque en un futuro puede incluir más funcionalidades como cancelar reservas, realizar búsquedas de otros servicios, etc.
- **ImaginBank:** Esta entidad ha creado un servicio de conversación automatizado disponible a través de la plataforma de mensajería Facebook Messenger para permitir a sus clientes encontrar descuentos y promociones exclusivos.
- **Moovit:** El chat en tiempo real está integrado en la plataforma de Facebook Messenger y permite a los usuarios buscar información sobre trayectos en redes de transporte público
- **Mahou:** Esta compañía ha incluido un chatbot en la plataforma de Facebook que se comunica con los clientes de manera natural e inmediata para ofrecerles actividades de ocio relacionadas con la cerveza.

2.2.2.4 AV en vehículos

Los coches conectados de diferentes fabricantes van adquiriendo mayores capacidades para una interacción máquina-persona mucho más natural, pero es necesario un paso que cree esa conexión. Es así cómo los asistentes virtuales entran en juego [10]. Algunos ejemplos son:

- **Volkswagen y Ford:** Estos fabricantes de coches incorporan en sus coches conectados a **Alexa**, el asistente virtual de Amazon, permitiendo que su cerebro de *deep learning* (conjunto de algoritmos de machine learning que pretenden modelar abstracciones de alto nivel en datos usando redes neuronales no lineales múltiples [11]) se encuentre al servicio de los vehículos y usuarios. Con él se pueden gestionar operaciones relacionadas con el cliente, como el control de música, gestión de correos, llamadas, etc.; así como aspectos relacionados con el coche como el estado del general del vehículo o su nivel de combustible, entre otras.
- **Nissan y BMW:** El asistente virtual de Microsoft, **Cortana**, es el sistema que incorporan los coches conectados de ambas compañías automotrices. Al igual que Alexa, este sistema permite a los usuarios acceder a su calendario, lista de tareas, correo y demás operaciones que se pueden realizar con la versión de Cortana para ordenador.
- **Hyundai:** Los coches de esta compañía que cuentan con el servicio Blue Link tienen compatibilidad inmediata con el asistente virtual de Google (**Google Assistant).** Dicho asistente permite, entre otras cosas, controlar la temperatura del automóvil, bloquear puertas, encender luces o tocar la bocina.

2.3 Tecnología utilizada

La meta es conseguir mantener una conversación entre un usuario y un asistente virtual, lo que significa automatizar una interacción sin perder la manera natural en que se comunican los seres humanos. Para conseguir esto, en este trabajo se utiliza una de las dos soluciones conversacionales que la empresa IBM ofrece al mercado: **Watson Conversations,** junto con unos servicios complementarios invocados a través de **API's Watson**. Esta tecnología de procesamiento de lenguaje natural unida a técnicas de inteligencia artificial y machine learning permiten construir una aplicación cognitiva para proporcionar soporte y ayuda tanto a usuarios finales cómo a agentes de un determinado sector comercial.

Como se menciona en el párrafo anterior, IBM ofrece dos soluciones conversacionales para el uso y creación de un AV. Antes de exponer las características de la opción utilizada en este proyecto, se va a comentar de forma breve la primera solución: **Watson Virtual Agent.** Es una solución pre-entrenada para una industria y empaquetada como un producto de software como servicio *(SaaS)* para usuarios de negocio. Permite una configuración rápida de agentes virtuales sin necesidad de habilidades técnicas especiales. Su aplicación es transversal a múltiples escenarios e incorpora contenido de servicio o atención al cliente final, con análisis enriquecidos en tiempo real de la relación con el cliente que ayuda a la empresa a mejorar y adaptarse constantemente a las necesidades de cada persona [12].

La otra solución de IBM que se utiliza en este proyecto son los servicios cognitivos Watson, que proporcionan API´s para clientes que deseen desarrollar sus propias aplicaciones conversacionales. Watson permite crear apps conversacionales en cualquier canal (por ejemplo: móvil, web, mensajería, robots, etc.) utilizando el entorno de desarrollo de diálogos y de entrenamiento **Watson Conversation** que permite a los expertos en cada dominio construir flujos de diálogo para asistentes cognitivos. Además de esto, IBM ofrece un conjunto de servicios que brindan al AV la capacidad de "escuchar" y de "hablar" mediante las API's de **Watson Speech to Text** y **Watson Text to Speech**, respectivamente. El uso de estas herramientas permite crear un asistente cognitivo con la capacidad de interactuar de forma conversacional, contextual y documentada con el usuario de modo que no solo entienda lo que está diciendo y cómo lo está diciendo (intención del usuario expresada de modo natural con distinto tono, emoción, etc.) sino que además aprenda sobre las preferencias y comportamiento del cliente y devuelva una respuesta correcta en el momento en que tenga más sentido y de la forma más adecuada.

En resumen, con estas capacidades, Watson permite satisfacer y gestionar de forma automatizada los requisitos del usuario mediante una interfaz conversacional y omnicanal, pudiéndose utilizar dentro de un sitio web con una interfaz conversacional, una ventana conversacional dentro de una aplicación móvil, un chatbot sobre una aplicación de mensajería o red social (Facebook Messenger, Slack, Telegram…), un canal telefónico de respuesta de voz interactiva (IVR), un dispositivo que escucha en un determinado lugar (un coche, una habitación, etc.) o incluso un robot que no sólo conversaría sino que, además, sería capaz de gesticular.

3 Diseño

En esta sección se va a detallar la forma en la que se ha elaborado el proyecto y el plan de desarrollo seguido. También se van a describir los distintos servicios que han sido necesarios para construir el agente virtual, así como la forma de interacción existente entre ellos, para más adelante, en la siguiente sección, mostrar la manera en la que se ha implementado este sistema.

De forma general, el proyecto se encuentra dividido en dos partes: *Interfaz* y *Servicio*, como se ilustra en la figura 3-1.

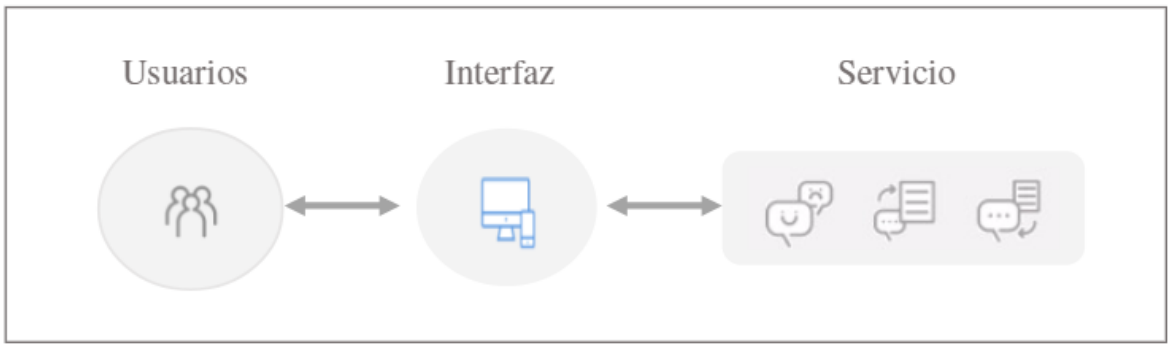


Figura 3-1: Esquema general del sistema.

La **Interfaz** es la parte del software que "interactúa" con los usuarios del sistema, recolectando los datos de entrada (como el texto o la voz) para enviarlos al **Servicio** que se encarga de procesar esta entrada y producir una respuesta que el Interfaz recibe y muestra al usuario.

El uso de este tipo de esquema viene motivado por la flexibilidad que presenta ya que ambas partes pueden ser desarrolladas con independencia una de otra únicamente teniendo en cuenta los servicios que el sistema ofrece. Por ejemplo, a la Interfaz no le hace falta conocer cómo está implementada la conversación en la parte del Servicio, solamente le hace falta saber cuál debe ser la entrada y cuál será la respuesta obtenida para utilizarlas adecuadamente. Ambas partes pueden verse entre sí como "cajas negras" con lo que se consigue que el sistema sea más fácil de entender ya que se da una visión más clara del conjunto, siendo más robusto y menos complicado de mantener pues en caso de ocurrir un fallo éste puede ser aislado y abordado ágilmente.

Ambas partes del sistema a su vez se dividen en módulos internos para realizar de manera eficiente las tareas que les corresponden. Una información más detallada de las funciones específicas de cada módulo, la forma en la que éstos están organizados y el plan seguido se muestran en las siguientes subsecciones.

3.1 Módulo Servicio

El Servicio, en este proyecto, es el área dedicada a la parte lógica del AV. Se encarga de proporcionar todas las funciones necesarias para que el sistema se ejecute adecuadamente brindando al usuario la sensación de estar interactuando con un agente real al que se puede dirigir de manera escrita o mediante voz, esperando una respuesta de la misma forma.

Para conseguir esta interacción, el asistente virtual debe ser capaz de cumplir con los principales requisitos, siendo estos:

- Procesamiento adecuado del lenguaje natural y sus expresiones para mantener una conversación coherente y responder a las preguntas y peticiones de los clientes, proporcionando información sobre los servicios que ofrece y realizando las acciones solicitadas.
- Procesamiento correcto de entrada y salida de voz, esto es, capacidad para mantener una comunicación oral con el usuario actuando como receptor y emisor a la vez.

La estrategia adoptada para cumplir estos requisitos se basa en la técnica "divide y vencerás" que implica resolver este problema (difícil a priori) dividiéndolo en partes más simples, pero correctamente funcionales, para luego integrarlas y conseguir el objetivo final.

Siguiendo esta técnica, este módulo queda subdividido en tres partes correspondientes a los requisitos antes descritos, tal y como se aparece en la Figura 3-2.

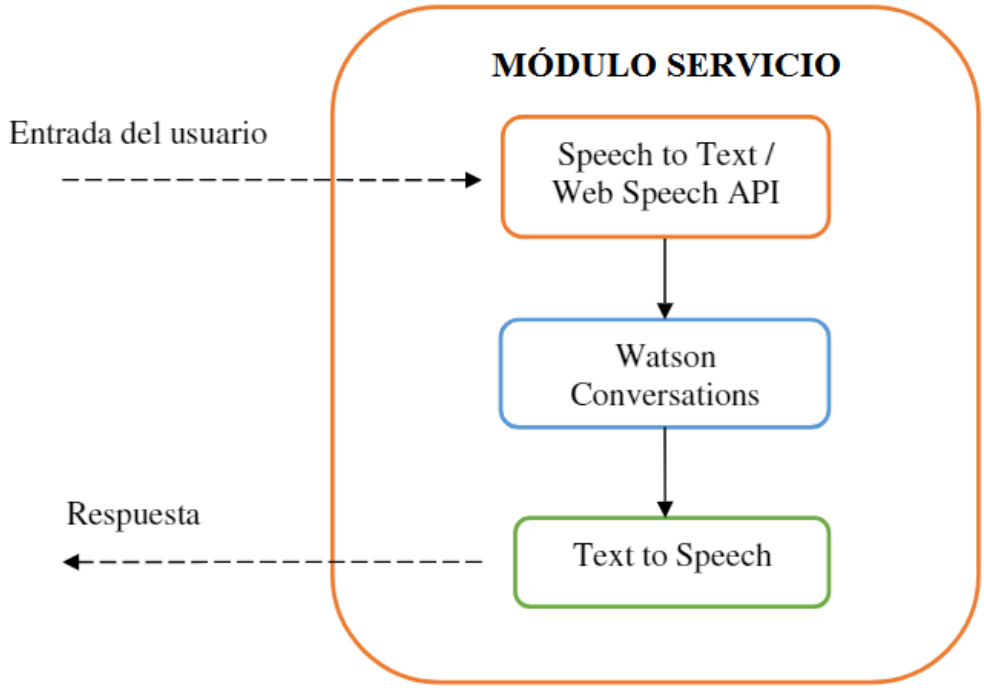


Figura 3-2: Sub-módulos del Servicio.

Los sub-módulos de la figura describen la forma en la que se procesa la petición del usuario para producir una respuesta junto con las herramientas software utilizadas para las tareas:

1. Procesamiento de entrada de voz: uso del servicio Speech To Text alojado en IBM Bluemix junto con uso de Web Speech API.
2. Procesamiento del lenguaje natural: uso del servicio conversacional Watson Conversations alojado en IBM Bluemix.
3. Procesamiento de salida de voz: uso del servicio de Text to Speech alojado en IBM Bluemix.

La plataforma Bluemix mencionada aquí se va a estudiar con más detalle en el capítulo de *Desarrollo*. Por ahora lo que hace falta sabe es que se trata de un entorno de plataforma como servicio (PaaS) desarrollado por IBM y que soporta varios lenguajes de programación y servicios que permiten crear, ejecutar, desplegar y gestionar aplicaciones en la nube[13].

Los apartados siguientes mostrarán la forma en la que trabajan estos servicios, cómo se han usado y la manera en la que se ha realizado la integración de todos ellos.

3.1.1 Procesamiento del lenguaje natural

La funcionalidad más importante de un asistente virtual es su capacidad para comprender el lenguaje natural, dar una respuesta con sentido y coherente adecuándose siempre al contexto de la conversación. Para dotar al AV de esta capacidad se hace uso del servicio **Watson Conversation** alojado en la plataforma IBM Bluemix. La plataforma proporciona un entorno de desarrollo online que permite construir la conversación en tres pasos: definiendo intenciones, definiendo entidades y elaborando un diálogo a partir de las definiciones anteriores.

Las **intenciones** representan los propósitos del usuario, más específicamente, las acciones que los usuarios desean realizar sobre la aplicación. Para definir las intenciones en el agente virtual de este proyecto el primer paso es entender la forma en la que se deben agrupar/dividir todas las posibles peticiones de un cliente y crear un grupo de una determinada temática teniendo en cuenta el área en que se enfoca el AV (panel de control de un coche). Por ejemplo, se puede crear un grupo para "Música" en donde se introducirán entradas del usuario del tipo "Apaga la música", "Reproduce música", etc. Este paso es muy importante para la aplicación ya que determina los flujos de diálogo que es necesario crear.

Una vez definidas todas las intenciones que un usuario podría necesitar, el siguiente paso es crear entidades. En la entrada de un usuario, una **entidad** representa un término u objeto que proporciona una aclaración o especifica el contexto de una determinada acción. Al igual que las intenciones representan verbos (acciones que el usuario desea realizar), las entidades representan sustantivos (el objeto o el contexto de una acción). El uso de entidades tiene una gran importancia para elaborar el diálogo porque gracias a ellas es posible que una misma intención pueda ser utilizada para indicar distintas acciones.

El concepto de entidad puede parecer algo confuso, pero es muy sencillo. Para entenderlo mejor se propone el siguiente ejemplo: supongamos que el usuario introduce a la aplicación las sentencias "enciende las luces del coche", "pon en marcha el aire acondicionado", "pon música". Estas oraciones representan una única intención "activar *algo*". Ese *algo* son las entidades que, en este ejemplo, son las luces del coche, el aire acondicionado y la música.

Cuando el usuario introduce una frase en la aplicación se reconocen tanto las intenciones como las entidades, por lo que un buen uso de ambas facilita mucho la construcción del diálogo.

El último paso para conseguir que el agente sea capaz de responder adecuadamente a cualquier petición viene determinado por la elaboración del flujo que seguirá la conversación una vez se inicie. Este flujo es a lo que se denomina **diálogo** dentro del servicio Watson Conversation y es el encargado de dar una respuesta basada en las intenciones y entidades que se identifiquen dentro de la entrada del usuario. Sin entrar en materia de implementación (tema que se verá ampliamente en el capítulo de *Desarrollo*) se puede decir que un diálogo está compuesto por nodos que representan una entrada

(intención o entidad) y que cada uno de estos nodos tiene una reacción determinada para dicha entrada, pudiendo ser una respuesta que se muestre al usuario o un paso para la evaluación de otro nodo.

Gráficamente un diálogo se muestra estructurado en forma de árbol compuesto por nodos. Al nodo principal o raíz se le denomina *Conversation Start*. Este nodo tiene "hermanos" que se ubican a continuación de él. Para que el diálogo sea completo cada nodo hermano representa una de las intenciones definidas, lo que permite recoger toda la información y acciones que el usuario pueda introducir y de esta manera dar una respuesta útil. Los nodos pueden tener distinto número de hijos dependiendo de la intención que se esté evaluando, cada uno con una condición que puede ser una de las entidades definidas previamente o cualquier expresión que al evaluarse devuelva verdadero o falso. Al igual que se tiene un nodo principal que se evalúa siempre que se inicia una conversación, también se puede tener un nodo final llamado por convenio *Anything Else* con el que se consigue que el diálogo devuelva una respuesta para cualquier entrada que llegue a ese nodo (véase Figura 3-3).

Un nodo puede estar asociado a una respuesta o, dependiendo de la condición que se evalúe, a una llamada a otro nodo. Esta llamada se denomina "salto" y se puede producir desde cualquier nodo (sea principal o hijo) hacia cualquier otro nodo.

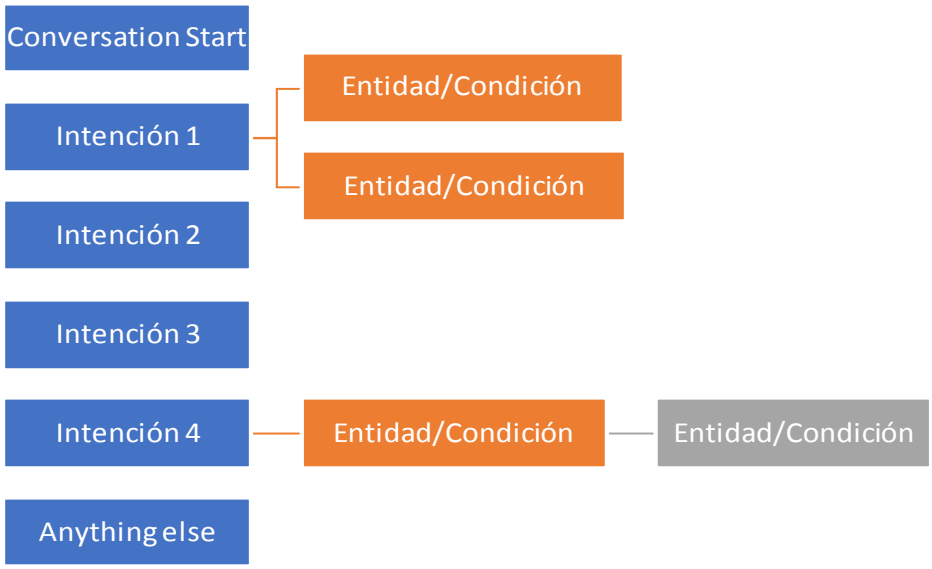


Figura 3-3: Esquema de estructura en árbol de un diálogo.

Antes de entrar en el proceso de evaluación, es necesario explicar lo que es un *nodo contextual* dentro de entorno de desarrollo: se denomina nodo contextual al último nodo que ha sido evaluado cuando el diálogo da una respuesta a una entrada del usuario. Este nodo contiene un identificador que se almacena en la propiedad *context.system.dialog_stack* y es a partir del cual se empieza la evaluación de la siguiente entrada. Al inicio de la conversación, se empieza evaluando el nodo *Conversation Start*.

El árbol se evalúa secuencialmente de arriba hacia abajo siguiendo el orden en que se encuentran los nodos. El proceso de evaluación consta de dos rondas:

- En la primera ronda, se trata de encontrar una respuesta dentro de los hijos del nodo contextual. Se comprueban las condiciones de cada hijo y si alguna de ellas es verdadera se devuelve la respuesta que resulta de la evaluación del nodo y se añade el identificador del primer nodo hijo como nodo contextual (si el nodo no tiene

hijos, se marca como nodo contextual el nodo raíz). Si no hay concordancia de ninguna condición de los nodos hijos, se pasa a la siguiente ronda de la evaluación.

- La segunda ronda, también llamada de nivel alto, evalúa los nodos hermanos al principal, empezando por la raíz, en busca de alguna concordancia para dar la respuesta.

Debido al flujo que sigue el árbol de diálogo, una parte esencial es organizar de manera correcta los nodos, empezando desde los que tienen condiciones más específicas y continuando con aquellos de condiciones más generales, para evitar que la conversación se quede estancada en un nodo cuya respuesta no es la esperada para la intención/entidad detectada en la entrada del usuario.

En la siguiente sección se detallan los pasos seguidos para la construcción del diálogo aplicando los conceptos aquí empleados.

3.1.2 Entrada y salida de voz

Una de las características más importantes de un asistente virtual es su capacidad para mantener una conversación oral con la persona que lo está utilizando actuando como receptor y emisor al mismo tiempo. Para dotar al AV de esta capacidad se hace uso de dos interfaces de programación de aplicaciones que permiten la transcripción de voz a texto, Web Speech API y Speech to Text API, junto con el servicio de IBM Watson para la transformación de texto a voz Text to Speech.

3.1.2.1 Entrada de voz

La primera herramienta se utiliza en el proyecto para llevar a cabo la transformación de voz a texto directamente desde la aplicación. **Web Speech API** es una librería JavaScript que permite el reconocimiento de voz en el navegador haciendo uso de los elementos hardware como puede ser el micrófono incorporado en un ordenador o dispositivo móvil o un micrófono externo por lo que cuando se ejecuta la aplicación se solicitan permisos al usuario para su uso. Esta API cuenta con una interfaz llamada *SpeechRecognition* cuyos métodos permiten escuchar la voz humana mediante micrófono, capturarla, procesarla y devolver una transcripción muy precisa con una latencia de espera relativamente baja. Los métodos mencionados y su forma de uso se explican en detalle en el capítulo de desarrollo.

Entre los servicios de Watson que proporciona IBM se encuentra **Speech to Text** para la trasformación de voz a texto, el cual se utiliza en la aplicación para permitir al usuario transcribir archivos de audio que se encuentren almacenados en el dispositivo en el que se esté ejecutando. El servicio detecta automáticamente los bits de audio y de manera interna hace uso de inteligencia artificial para combinar información sobre la estructura de la gramática y el idioma con los conocimientos de la composición de la señal de audio, generando de esta manera una transcripción textual precisa.

Para la transcripción el servicio acepta los siguientes formatos de audio: *audio/flac, audio/l16, audio/wav, audio/ ogg, audio/webm, audio/mulaw, audio/basic*. Sin embargo, es recomendable utilizar archivos en formato *audio/flac* o *audio/ogg* debido a que ofrecen una mejor compresión de los datos y el límite del servicio es de 100MB por audio a transcribir.

Aunque en este proyecto se utilizan ficheros de audio en castellano, la Figura 3-4 muestra los idiomas y tipos de archivo soportados actualmente.

Model Name	Language	Minimum Sampling Rate
ar-AR_BroadbandModel	Modern Standard Arabic	16 KHz
en-UK_BroadbandModel	UK English	16 KHz
en-UK_NarrowbandModel	UK English	8 KHz
en-US_BroadbandModel (the default)	US English	16 KHz
en-US_NarrowbandModel	US English	8 KHz
es-ES_BroadbandModel	Spanish	16 KHz
es-ES_NarrowbandModel	Spanish	8 KHz
fr-FR_BroadbandModel	French	16 KHz
ja-JP_BroadbandModel	Japanese	16 KHz
ja-JP_NarrowbandModel	Japanese	8 KHz
pt-BR_BroadbandModel	Brazilian Portuguese	16 KHz
pt-BR_NarrowbandModel	Brazilian Portuguese	8 KHz
zh-CN_BroadbandModel	Mandarin Chinese	16 KHz
zh-CN_NarrowbandModel	Mandarin Chinese	8 KHz

Figura 3-4: Lenguajes y modelos soportados por Speech to Text

3.1.2.2 Salida de voz

Para dotar al AV de la capacidad de "hablar" se utiliza el servicio Text to Speech de IBM Watson Services. Este servicio se ejecuta inmediatamente después de recibir la respuesta del diálogo Watson Conversation visto anteriormente y da como resultado un fichero de audio configurable para obtener un formato específico (soportando los mismos formatos que el servicio Speech to Text).

El servicio ofrece una serie de características que se pueden aplicar al audio de salida para que se escuche "natural", dejando a un lado la clásica voz de robot, monótona y carente de sentimiento:

- **Voces:** El texto se puede sintetizar para obtener una salida en 6 idiomas: español, inglés, alemán, francés, portugués, japonés e italiano. Para cada uno de ellos se cuenta con una voz masculina o femenina y un dialecto propio, lo que se traduce en cambios de entonación, acento y expresión. Para el idioma español existen cuatro voces y tres dialectos distintos ilustrados en la Tabla 3-1.

VOZ	Enrique	Laura	Sofía	Sofía
GÉNERO	Masculino	Femenino	Femenino	Femenino
DIALECTO	Castellano	Castellano	Latinoamericano	Norteamericano

Tabla 3-1: Voces de Text to Speech para español.

- **Expresividad:** El servicio acepta texto plano o también texto marcado con SSML (Speech Synthesis Markup Language, un lenguaje basado en XML que proporciona anotaciones de texto para aplicaciones de síntesis de voz [14]) que se puede modificar para que la voz de salida exprese la frase o palabra con tono de buenas noticias (*GoodNews)*, disculpas (*Apology*) o incertidumbre (*Uncertainty*).
- **Transformación de voz:** También se puede extender el SSML que marca un texto para modificar el tono, la respiración, el ritmo, la velocidad o el timbre de la voz de salida.
- **Personalización:** El servicio Text to Speech también ofrece una interfaz de personalización que permite especificar cómo pronunciar palabras inusuales para el idioma con el que se esté trabajando. La pronunciación se puede definir de dos maneras, haciendo uso del Alfabeto Fonético Internacional (IPA) o con la Representación Fonética de Símbolos de IBM (SPR).

Los usos de las características mencionadas de este servicio en la aplicación se comentarán en la siguiente sección.

3.1.3 Integración de los servicios

Ahora que se han visto los servicios que intervienen en la creación del agente virtual, el siguiente paso es integrarlos, es decir, enlazar cada uno de ellos para conseguir el objetivo de la aplicación: que sea capaz de escuchar la voz de un usuario y transcribir lo que diga a palabras, utilizar el texto obtenido como entrada al servicio Watson Conversation para iniciar un diálogo y una vez obtenida la respuesta en forma de texto transformarla a voz mediante el servicio Text to Speech.

Como elemento integrador de todos estos componentes se utiliza **Node-Red,** una herramienta de programación de código abierto útil para conectar dispositivos hardware, APIs y servicio en la nube. Está construido sobre Node.js que, aprovechando al máximo su modelo de eventos no bloqueantes, lo hace ideal para correr en la red y desplegarse de manera eficiente en hardware de bajo costo y también en la nube [15].

Esta herramienta está especialmente orientada para el Internet de las Cosas (IoT) y proporciona un entorno de diseño gráfico de flujos de integración accesible desde un navegador (véase Figura 3-5).

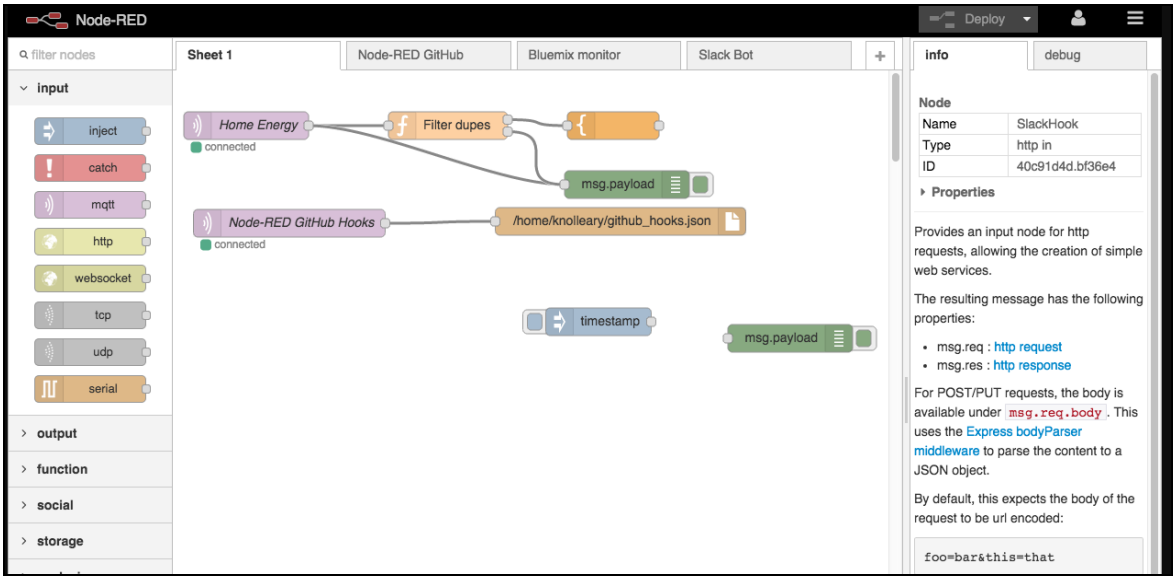


Figura 3-5: Editor de flujos Node-Red

En el proyecto, Node-Red se ha utilizado de dos maneras, una instalando localmente la herramienta en el ordenador y otra utilizando un servicio accesible desde la red y alojado en IBM Bluemix. En cualquier caso, la forma en la que se crea y despliega un flujo es la misma:

1. Se seleccionan los nodos que van a intervenir en la aplicación. La herramienta permite instalar nodos que corresponden a todos los servicios de Watson de una manera sencilla a través del *Manage Palette* disponible en el editor (paquete *node-red-node-watson*).
2. Una vez se han seleccionado los nodos se deben conectar de la manera en que se va a utilizar el servicio. Para conectar nodos basta con unir los extremos de uno y otro nodo.
3. En este punto, los nodos se encuentran únicamente en el editor por lo que se deben desplegar al servidor. Esto se consigue haciendo click en el botón *Deploy*.
4. Después del depliegue, la funcionalidad del flujo queda totalmente accesible.
5. De manera opcional, la herramienta permite exportar el flujo creado, guardado los datos de los nodos y sus conexiones en un archivo JSON. De la misma forma permite importar un archivo JSON y reconstruir el flujo que representa.

En el caso del AV derrollado, siguiendo estos pasos, se pueden unir los tres servicios fácilmente a través de sus nodos y desplegar la aplicación para que sea accesible a través de la red. Más adelante se verán los pasos seguidos para construir el flujo que propociona toda la funcionalidad que se ha mencionado en esta sección.

3.2 Módulo Interfaz

Este módulo hace referencia a la parte visual del proyecto, en otras palabras, hace referencia a la pantalla desde la que el usuario interactúa directamente con el asistente virtual manteniendo una conversación. Sirve de conector entre el usuario y el módulo Servicio, por lo que debe permitir que todas las acciones de comunicación oral y escrita del AV sean accesibles para el usuario de una manera fácil e intuitiva.

Como se ha explicado en el aparatado anterior, se utiliza la herramienta Node-Red para ejecutar el servicio conversacional del asistente virtual. Cuando se despliega el flujo creado toda la funcionalidad se encuentra accesible y puede ser utilizada a través de la red haciendo uso de un nodo que crea peticiones HTTP GET o POST en el servidor de Node-Red. Este nodo, conectado a un nodo llamado *Template* (que toma directivas HTML válidas) permite crear una interfaz web dinámica que se ejecuta en la instancia de Node-Red.

Para crear la página Web se hace uso de la actual tecnología HTML5 para maquetar su contenido junto con CSS3 para codificar su diseño y estilo. Como ayuda se utiliza Bootstrap, un framework HTML, CSS y JavaScript para diseño web adaptativo, que permite que la página web desarrollada se ajuste a las dimensiones y características de pantalla del dispositivo en el que se visualiza, como pueden ser smartphones, tablets, ordenadores portátiles, etc., reduciendo el coste de desarrollo comparado con implementar la aplicación para distintos dispositivos[16]. Para permitir la interacción entre el usuario y la conexión al módulo Servicio se utiliza el lenguaje de programación JavaScript y como ayuda la librería JQuery que facilita el acceso a los elementos del documento html y

permite ejecutar peticiones AJAX [17] para intercambiar datos con el servidor sin tener que recargar toda la página web cada vez que se envía o recibe un elemento.

3.2.1 Sitio Web

En la Figura 3-6 se muestra el mapa del sitio web y a continuación, una breve explicación de las pantallas disponibles, la función que desempeña cada una y la navegación existente entre ellas.

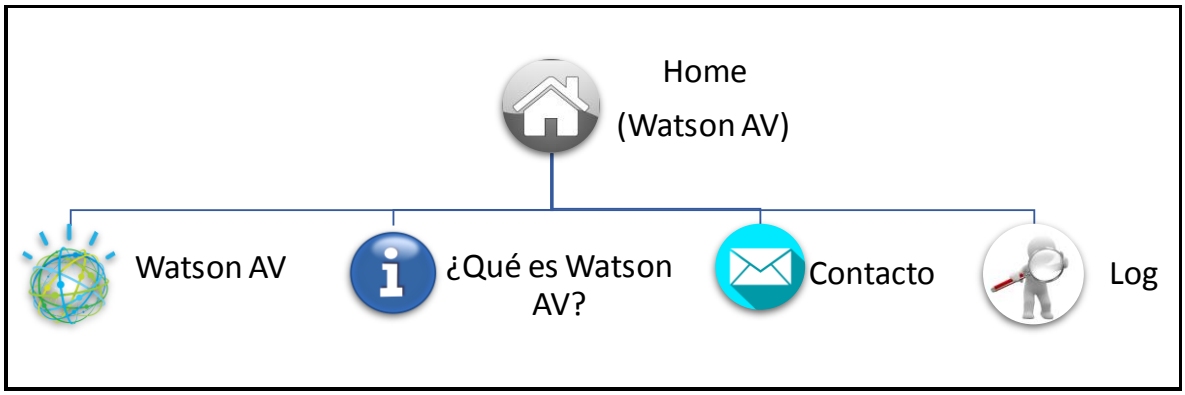


Figura 3-6: Watson AV. Organización del sitio web

El sitio web consta de 4 pantallas:

1. **Watson AV:** Es la pantalla en la que el usuario se puede comunicar directamente con el asistente virtual a modo de conversación. Consta de una región de mensajería en donde se visualizan las peticiones del usuario y también la respuesta del AV (esta respuesta también se ejecuta en forma de audio). Además, cuenta con un par de botones para permitir al usuario realizar la conversación de manera oral mediante el micrófono o subir un archivo de audio para que el asistente lo interprete. Es la pantalla inicial de la aplicación.
2. **¿Qué es Watson AV?**: En esta pantalla se muestra información sobre las capacidades y características con que cuenta el asistente virtual, así como los servicios que ofrece.
3. **Contacto**: Aquí se muestra información de contacto de las personas involucradas en el proyecto (alumno, tutor y ponente).
4. **Log**: Página en la que se muestran las acciones que se ejecutan en la conversación con el asistente virtual y también los posibles errores que se puedan presentar. Por ejemplo, un error puede ser el de intentar enviar al AV un archivo inexistente o en un formato no permitido. En la versión final no es visible para el usuario, pero se puede activar por el desarrollador para realizar debug.

En el Anexo I se encuentra las maquetas utilizadas para el diseño de la aplicación web.

3.3 Plan de desarrollo

Teniendo en cuenta la funcionalidad que el proyecto debía adquirir y la magnitud de éste, se dividió el problema en varios apartados y se llevó a cabo una estimación de tiempo inicial para la preparación y elaboración de cada tarea.

La Tabla 3-2 muestra las actividades en las que se dividió el problema inicial para poder construir el Agente Virtual de una manera incremental. Cada actividad viene

acompañada del ámbito al que pertenece (Interfaz o Servicio) así como una duración (en semanas) estimada de acuerdo a la magnitud de cada una.

Ámbito	Actividad	Duración
	Desarrollo Primer Piloto:	
Servicio	Conversations (investigación y desarrollo)	3
Servicio	Speech to Text (investigación y desarrollo)	1
Servicio	Text to Speech (investigación y desarrollo)	1
Servicio	Node-Red (investigación)	3
Servicio	Flujos de Node-Red con Watson Services (desarrollo)	2
Servicio	Pruebas	2
Interfaz	**Fin Primer Piloto**	1
Interfaz	**Segundo Piloto**	2
TOTAL		15

Tabla 3-2: Cronograma y estimación inicial de actividades

Como se puede observar en la tabla, las primeras doce semanas se estimaron para la realización de la parte Servicio del proyecto tras lo cual se contaría con una primera versión del Agente Virtual aún sin interfaz gráfica. La siguiente semana se planificó para el desarrollo de una parte visual muy simple, para luego perfeccionarla en las dos siguientes semanas siguiendo las maquetas diseñadas para el sitio web.

Esta tabla se intentó seguir de la manera más fiel posible, pero según se avanzaba en el proyecto hubo algunas modificaciones temporales debido a la sobreestimación y la subestimación de algunas de las tareas. La Tabla 3-3 muestra el cronograma final de actividades.

Ámbito	Actividad	Duración
	Desarrollo Primer Piloto:	
Servicio	Conversations (investigación y desarrollo)	3
Servicio	Speech to Text (investigación y desarrollo)	½
Servicio	Text to Speech (investigación y desarrollo)	½
Servicio	Node-Red (investigación)	2
Servicio	Flujos de Node-Red con Watson Services (desarrollo)	4
Interfaz	**Fin Primer Piloto**	1
Interfaz	**Segundo Piloto**	2
Interfaz / Servicio	Pruebas	1
	Validación y finalización	1
TOTAL		15

Tabla 3-3: Cronograma final de actividades.

Se puede ver que el estudio de los servicios Speech to Text y Text to Speech tuvieron menor duración que la estimada, mientas que la integración de estos servicios con flujos de Node Red tuvieron una mayor duración. Otro cambio ocurrido tuvo lugar en las pruebas, que se desarrollaron al final del desarrollo de los pilotos para abarcar tanto la componente de Servicio como la componente de Interfaz, y una etapa de validación y corrección de fallos encontrados en las pruebas.

4 Desarrollo

En la sección anterior se explicó la manera en la que está organizada la aplicación, los módulos y sub-módulos que la componen y la función que realiza cada uno para constituir el asistente virtual Watson AV.

Teniendo en cuenta estos aspectos, en esta sección se va a explicar cómo se desarrollan e implementan los módulos Servicio e Interfaz, adentrándose en aspectos técnicos de programación, así como en la forma en que se crean los servicios *Watson Conversation*, *Speech to Text*, *Text to Speech* y también la instancia de *Node-Red* dentro de la plataforma IBM Bluemix.

Antes de empezar se va a hacer una breve introducción a la plataforma Bluemix y a la manera en la que se crean y utilizan los sistemas que se encuentran disponibles en ella, centrando la atención en los servicios cognitivos Watson que interviene en este proyecto.

4.1 Introducción a IBM Bluemix

Bluemix es un entorno de plataforma como servicio basado en Cloud-Foundry[1] y desarrollada por IBM. Proporciona acceso inmediato a diversos sistemas y servicios que permiten crear, ejecutar, desplegar y gestionar aplicaciones en la nube. Entre ellos se incluyen servicios DevOps (para despliegue y desarrollo de aplicaciones), **servicios cognitivos Watson**, servicios para el desarrollo y mantenimiento de aplicaciones móviles, para el análisis de datos y otros muchos [18].

La Figura 4-1 muestra los servicios disponibles en la plataforma.

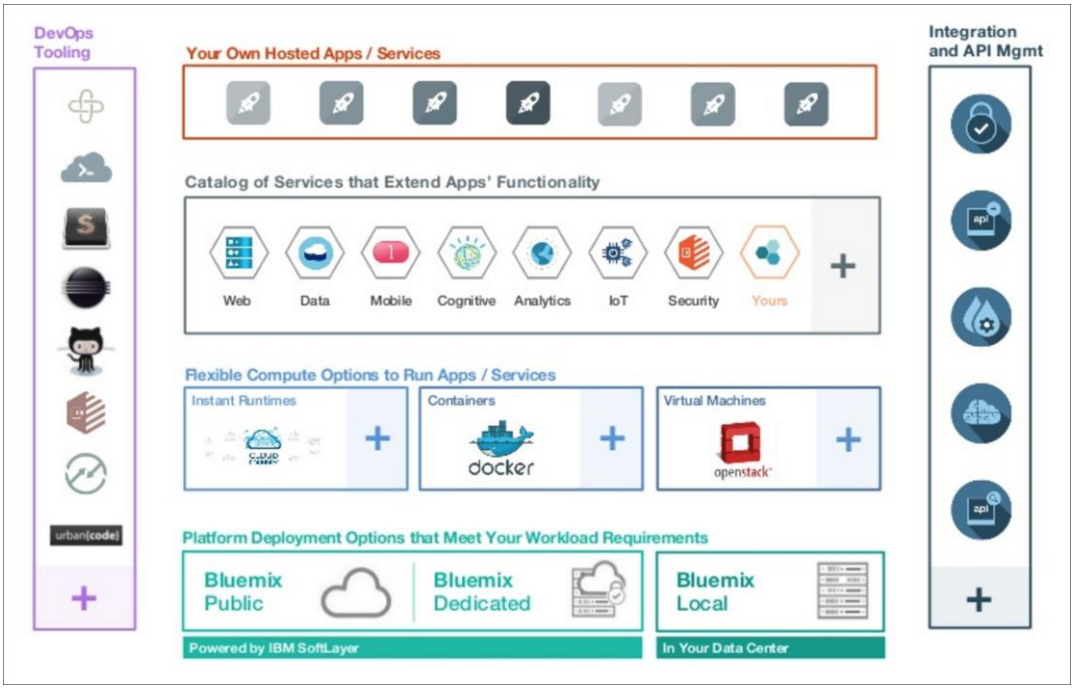


Figura 4-1: Servicios de la plataforma Bluemix

[1] Cloud-Foundry es el estándar de código abierto para PaaS. Para conocer más detalles de esta plataforma: http://docs.cloudfoundry.org/

De entre los servicios del catálogo de Bluemix se utilizan en este proyecto tres: Watson Conversation, Speech to Text y Text to Speech, que se enmarcan dentro del grupo *Watson Cognitive Services.* Además de estos servicios, Bluemix proporciona servidores en los que se permiten desplegar y ejecutar aplicaciones propias de la plataforma o también apps pertenecientes a terceros, como es el caso de la aplicación Node-Red utilizada para la integración de los servicios.

Cada vez que se crea un servicio, la plataforma le asigna automáticamente unas **credenciales.** Estas credenciales se muestran en formato JSON y contienen toda la información que una aplicación necesita para conectarse a dicho servicio. Para los tres servicios antes mencionados, las credenciales tienen un formato común que incluye su dirección URL, un nombre de usuario y una contraseña únicos e inmutables, como se aprecia en la Figura 4-2.


```
{
  "url": "https://stream.watsonplatform.net/speech-to-text/api",
  "username": "6f00ccbb-4af3-4bae-a144-47636b9822f9",
  "password": "KtF1E5cwBhte"
}
```


Figura 4-2: Ejemplo de credenciales de servicio Speech to Text

Ahora que se conoce de manera general en qué consiste y cómo funciona la plataforma de Bluemix, en la siguiente subsección se va a explicar la forma en la que los tres servicios aquí mencionados se utilizan, primero individualmente y luego en conjunto, para constituir el módulo Servicio.

4.2 Módulo Servicio

En esta sección se va a estudiar la manera en la que se han creado y utilizado los tres servicios de Bluemix que proporcionan la funcionalidad de procesamiento del lenguaje natural y de entrada y salida de voz.

Aunque la aplicación Node-Red no pertenece a IBM y se puede ejecutar localmente fuera de Bluemix, también se va a comentar el uso de la herramienta dentro de la plataforma y cómo se puede acceder remotamente a ella para crear y modificar flujos o desplegar la aplicación.

4.2.1 Watson Conversation

Como se vio en el diseño, el uso de este servicio permite "generar" una conversación entre el asistente virtual y el usuario siguiendo tres pasos: definición de intenciones, definición de entidades y construcción del diálogo.

Intenciones

Para definir las intenciones con que cuenta el servicio, hace falta conocer cuáles pueden ser las posibles entradas del usuario teniendo en cuenta el ámbito en el que se va a desplegar el asistente virtual, que es el panel de control de un coche eléctrico. Aparte de las intenciones de ámbito específico también se definen intenciones de ámbito general, es decir, acciones que un usuario pueda realizar independientemente del área al que se dedica el asistente virtual, como pueden ser las acciones de saludo o despedida.

Se recuerda que las intenciones son sinónimos de **acciones o reacciones** que el usuario puede indicar, pero lo puede hacer de muy distintas maneras por lo que hace falta agrupar aquellas que representen lo mismo.

Una vez se categorizan las intenciones, el siguiente paso es incluir **ejemplos** en cada clase que ayuden a identificar a qué intención pertenecen (la plataforma permite crear hasta 25,000 ejemplos), cuantos más ejemplos se proporcionen para una intención, ésta tiene más probabilidad de ser reconocida por el servicio. Teniendo en cuenta que los seres humanos a veces cometemos errores léxico/gramaticales, entre los ejemplos de una determinada intención también se deben incluir varios que presenten errores de este tipo de tal manera que unidos a ejemplos bien formados faciliten el reconocimiento de la frase, acción o cuestión que el usuario introduzca en la aplicación. Cada vez que se añade una intención o un ejemplo, el servicio Conversation se "entrena" para reconocer el nuevo dato.

Bluemix contiene un entorno de desarrollo que permite realizar estas acciones de una manera sencilla, además de facilitar ciertas modificaciones sobre las intenciones y ejemplos creados, como son renombrar y eliminar intenciones o editar, eliminar y mover de una intención a otra los ejemplos, como puede apreciarse en la Figura 4-3. El entorno también permite la opción de **importar** ejemplos junto con sus correspondientes entidades, para ello, cada ejemplo debe estar correctamente encapsulado dentro de un archivo CSV siguiendo el formato:

example, intent1
example, intent1
example, intent2

Figura 4-3: Interfaz de intenciones en el entorno de desarrollo.

En el Anexo II se puede ver las intenciones definidas para el proyecto.

Entidades

Aunque no es imprescindible declarar entidades, el uso de ellas ligado a intenciones ayuda a crear un mejor flujo en el diálogo de la conversación. Como explicó anteriormente, las entidades representan un sustantivo sobre el que se aplica una acción o, lo que es lo mismo, sobre el que se aplica una intención. Al tener varias clases de intenciones, todas ellas se pueden aplicar sobre distintos objetos, por lo que es necesario definir también distintas categorías para las entidades.

El proceso de definición de entidades es análogo al de intenciones: se identifican adecuadamente las categorías de las entidades teniendo en cuenta el ámbito en el que se desenvuelve la aplicación y para cada una de ellas se proporcionan valores de elementos pertenecientes a dicha categoría. Debido al lenguaje, un sustantivo se puede expresar de

varias formas a través de sinónimos, por lo que, para definir adecuadamente las entidades, junto a cada valor presente en una categoría se indica también un sinónimo o sinónimos que se puedan utilizar. El servicio permite incluir un máximo de 100,000 valores por categoría y también 100,000 sinónimos para cada valor.

Al igual que con las intenciones, la plataforma de Bluemix permite eliminar o modificar los nombres de los grupos, los valores y los sinónimos de cada entidad en el entorno de desarrollo online (como se aprecia en la Figura 4-4), aunque también se pueden importar entidades que se encuentren almacenadas en archivos tipo CSV, siguiendo el siguiente formato para definirlas: *entidad, valor, sinónimos.*

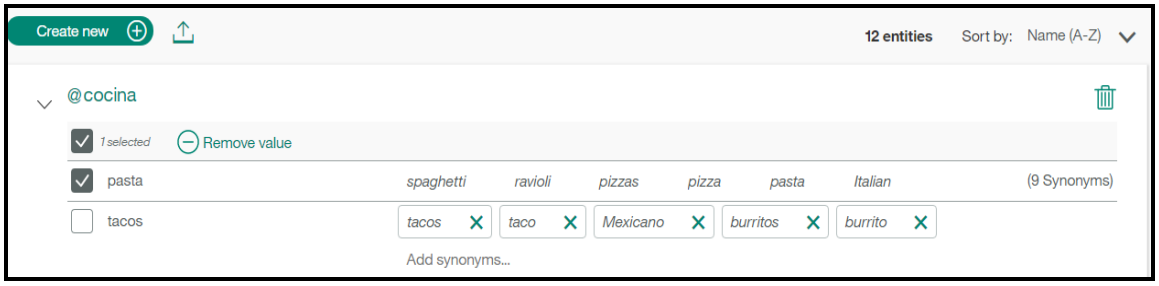


Figura 4-4: Interfaz de entidades en el entorno de desarrollo.

En el Anexo II se puede ver las entidades definidas para el proyecto.

Diálogo

Habiendo definido intenciones y entidades, el paso final es la construcción del flujo de diálogo que va a seguir la conversación. Es la parte más crítica ya que todo el sistema depende de que el servicio reconozca correctamente las distintas entradas del usuario y responda de una manera adecuada a ellas.

Para asegurar que todas las posibles entradas del usuario (las que han sido contempladas en este sistema) se capturen se crean los nodos teniendo como condición de evaluación a las intenciones definidas, por lo que se declaran tantos nodos como intenciones, en este caso 25. Como se mencionó en el diseño, aparte de estos nodos también hace falta crear dos adicionales, un nodo que se evalúa siempre al inicio de la conversación llamado c*onversation_start* y un nodo que se evalúa cuando no se reconoce la entrada llamado *anything_else*. En total se utilizan 27 nodos de los 25,000 que el servicio pone a disposición. Cada nodo debe contener una respuesta que se muestre cuando éste sea evaluado. En ocasiones las respuestas requieren acciones de mayor complejidad que únicamente devolver un texto, por ejemplo, para la entrada "Enciende las luces", el diálogo (y dentro de él el nodo evaluado) aparte de responder con un texto del tipo "Voy a encender las luces" también debe realizar dicha acción o, en el caso de que ya estén encendidas, decir "Las luces ya están encendidas". Para poder realizar una acción de este tipo se debe tener constancia de toda la conversación y todas operaciones realizadas hasta el momento. A esta constancia en la conversación se denomina **contexto**, en el que se almacenan todas aquellas variables que pueden cambiar durante el diálogo.

El entorno de desarrollo proporciona dos formas de incluir respuestas a los nodos, la primera está pensada para las respuestas simples, es decir, que únicamente devuelvan un texto, y la segunda está pensada para aquellas que modifican el contexto. Las Figuras 4-5 y 4-5 ilustran ambas formas para una misma respuesta.

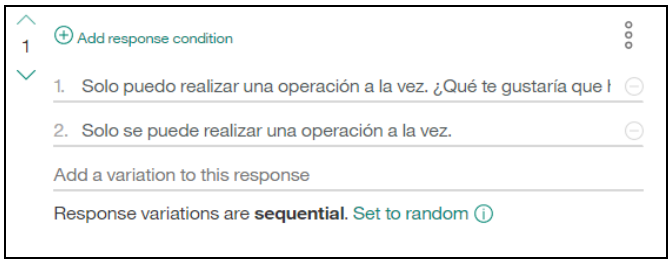


Figura 4-5: Ejemplo de edición simple.

```
{
  "context": {},
  "output": {
    "text": {
      "values": [
        "Solo puedo realizar una operación a la vez. ¿Qué te gustaría que haga?",
        "Solo se puede realizar una operación a la vez."
      ],
      "selection_policy": "sequential"
    }
  }
}
```

Figura 4-6: Ejemplo de edición avanzada.

En este documento no se va a explicar la forma en que se ha elaborado la respuesta de cada nodo, sino únicamente de aquellos que tienen más relevancia y cierta "complejidad" en el diálogo. Se entiende por "nodos complejos" a aquellos en los que no se devuelve una respuesta inmediatamente al ser evaluados, sino que incorporan otras operaciones dentro, como pueden ser cambios de estado o contexto en la conversación. El siguiente esquema (Figura 4-7) muestra dichos nodos:

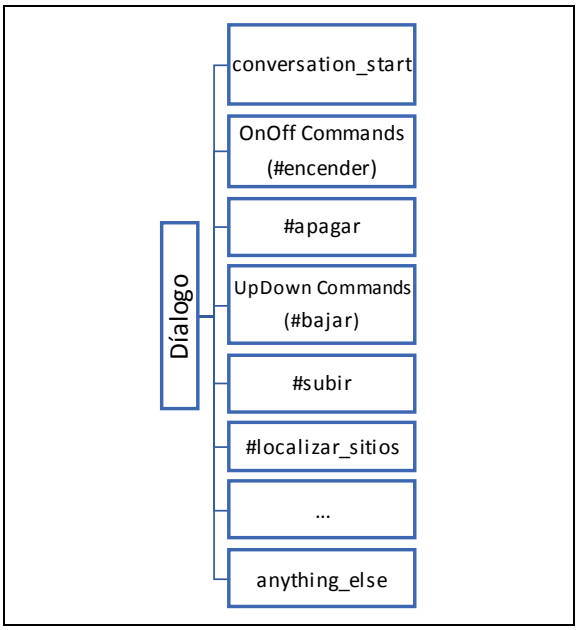


Figura 4-7: Esquema de nodos en el proyecto.

1. **Conversation_Start:** Este nodo devuelve una respuesta bienvenida a la conversación e inicializa el contexto, como se muestra en la Figura 4-8.

```
{
  "context": {
    "AConoff": "off",
    "appliance": "",
    "lightonoff": "off",
    "musiconoff": "off",
    "appl_action": "",
    "heateronoff": "off",
    "temperature": 20,
    "volumeonoff": "off",
    "wipersonoff": "off",
    "default_counter": 0,
    "max_temperature": 30,
    "min_temperature": 10
  },
  "output": {
    "text": {
      "values": [
        "¡Hola! Mi nombre es Watson, soy tu asistente virtual y estoy aquí para ayudarte en el viaje a bordo de este coche. Hoy parece un gran día para conducir. ¿Qué te gustaría que haga? "
      ]
    }
  }
}
```


Figura 4-8: Conversation_start: Respuesta avanzada

2. **OnOff Comands:** Este nodo se evalúa al detectar la intención #encender. Modifica la variable "appl_action" del contexto de "off" a "on", lo que quiere decir que la acción que se va a realizar sobre una entidad de tipo @dispositivo o @genero (música) es la de activar. Luego de modificar el contexto, este nodo "salta" a otra serie de nodos (hijos al nodo OnOff Commands) donde se evalúa la entidad introducida por el usuario.
   Ejemplo (Figura 4-9): Para la entrada del usuario "Encender luces" se reconoce la intención *#*encender y la entidad *@dispositivo:luces*, y con ello se modifica el contexto *appl_action = "on"* y *lightonoff = "on"*.

Figura 4-9: Respuesta para "Encender luces"

3. **#apagar:** La evaluación de este nodo modifica el contexto y pone *appl_action = "off"*. Después de esto "salta" al conjunto de nodos de *OnOffCommands* antes mencionados para evaluar la entidad reconocida y realizar la acción de apagar.

4. **UpDownCommands:** Este nodo reconoce la intención *#bajar* y modifica el contextp *appl_action = "decrease".* A continuación, hace una evaluación similar a la del nodo *OnOffCommands* para realizar la acción sobre la entidad que se reconozca en la entrada del usuario.
5. **#subir:** El comportamiento de este nodo es análogo al nodo *#bajar*, con la diferencia que se modifica el contexto *appl_action* dándole un valor de "*increase".*
6. **#localizar_sitios:** Este nodo se dispara al reconocer la intención *#localizar_sitios*. Luego mediante sus nodos hijos comprueba si en la entrada se reconoce alguna entidad de tipo *@confort,* en cuyo caso devuelve una respuesta adecuada a la entrada.

Si en la entrada del usuario se reconoce una intención, pero no una entidad que lo acompañe, los nodos comentado anteriormente utilizan la variable de contexto *repromt = true* para preguntar, en la siguiente evaluación, por una entidad en concreto, como el ejemplo que se muestra en la Figura 4-10: se busca un restaurante, pero no se indica de que tipo.

```
{
  "context": {
    "reprompt": true
  },
  "output": {
    "text": {
      "values": [
        "Por supuesto. ¿Tienes alguna comida específica en
mente?"
      ]
    }
  }
}
```

Figura 4-10: Uso de la variable de contexto *reprompt.*

El resto de nodos no comentados aquí siguen un esquema de uso similar a los anteriores, algunos modificando el contexto de acuerdo a las entradas y otros simplemente devolviendo una respuesta.

Para finalizar este apartado, hace falta mencionar que internamente toda la información sobre las intenciones, entidades y diálogo creados se almacenan en un archivo tipo JSON que puede ser **exportados** en cualquier momento para su visualización o edición de manera local. Esto permite, entre otras cosas, llevar un control sobre los cambios realizados exportando el archivo cada vez que se realice una alteración importante. En el caso de existir algún error o pérdida si se realiza una modificación en alguna parte del servicio, el uso de los archivos JSON permite tener una especie de *backup* para restaurar los datos, acto que se realiza a través de la opción **importar** (también disponible en el entorno de desarrollo) que carga todos los datos que se encuentren en almacenados en un archivo JSON y los muestran en el entorno de Bluemix.

4.2.2 Text to Speech y Speech to Text

Estos servicios, a diferencia de Watson Conversation, no proporcionan un entorno de desarrollo. Para utilizarlos, simplemente hace falta crear los servicios dentro de la plataforma y obtener sus credenciales. La configuración y utilización de ellos se va a comentar en el siguiente apartado.

4.2.3 Node-Red

Esta aplicación, como se comentó anteriormente, se puede utilizar de manera local o también dentro de la plataforma de Bluemix. La creación de una instancia de la aplicación en Bluemix permite el despliegue de la herramienta en un servidor gestionado por IBM, al que se accede mediante una dirección URL proporcionada por la plataforma. Para acceder a cualquier flujo HTTP creado, basta con añadir la ruta de ese flujo: URL/ruta

Independientemente de donde se despliegue, la forma de utilización y de creación de flujos es la misma. Dentro del editor se crean 4 flujos, uno para cada servicio de Watson utilizado y otro en el que se integra la página web de la interfaz gráfica, como se muestra en las Figuras 4-11, 4-12, 4-13 y 4-14.

Flujo Watson Conversation

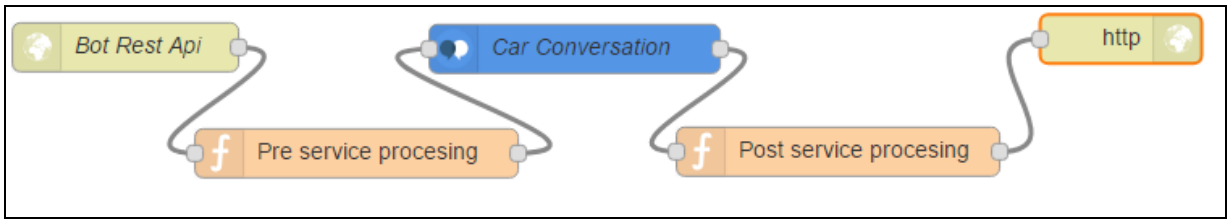


Figura 4-11: Flujo REST Watson Conversation.

- **Bot Rest Api:** Nodo HTTP de entrada, se configura con el método de acceso POST y la ruta /watsontalk/botchat.
- **Pre service procesing:** Función JavaScript de pre-procesamiento, toma los datos de entrada con los que ha llamado a la dirección HTTP y almacena el cuerpo del mensaje (la entrada del usuario) y el contexto de evaluación.
- **Car Conversation:** Este nodo se configura utilizando las credenciales del servicio Watson Conversation creado.
- **Post service procesing:** Función JavaScript de post-procesamiento que configura la respuesta de la invocación incorporando el contexto de la conversación y la respuesta devuelta por el servicio.
- **HTTP response:** Devuelve la respuesta de la invocación.

Flujo Text to Speech

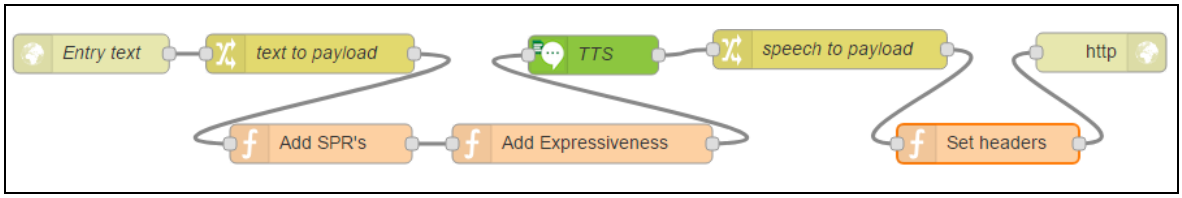


Figura 4-12: Flujo REST Text To Speech

- **Entry text:** Nodo HTTP de entrada que se configura con el método de acceso GET y la ruta **/watsontalk/sayit**
- **Text to payload:** Copia el valor de la variable *text_to_say* que se recibe mediante GET y lo almacena en la variable *msg.payload*.
- **Add SPR's:** Añade al texto de entrada notación SPR para modificar la pronunciación de ciertas palabras en el servicio.

- **Add Expressiveness:** Añade algunos tags SSML para mejorar la expresividad del texto de entrada.
- **TTS:** Nodo del servicio Text to Speech, se configura utilizando las credenciales del servicio creado.
- **Speech to payload:** Guarda en la variable msg.payload el audio producido por el servicio TTS.
- **Set headers:** Añade cabeceras que identifican el formato de audio para que se reconozca la salida.
- **HTTP response:** Devuelve como respuesta a la invocación un archivo de audio.

Flujo Speech to Text

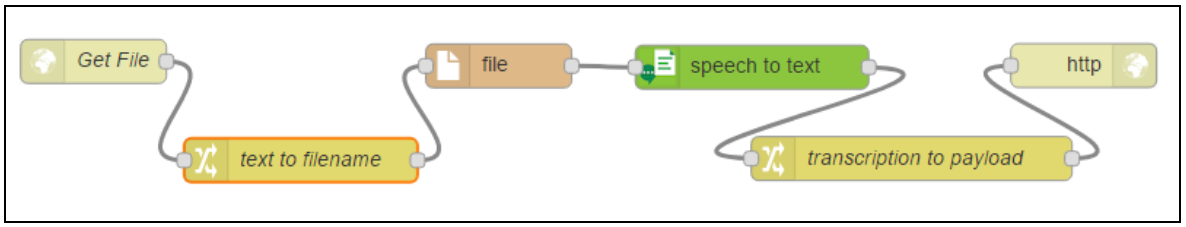


Figura 4-13: Flujo REST Speech to Text

- **Get File:** Nodo HTTP de entrada que se configura con el método de acceso GET y la ruta **/ watsontalk/getFile.**
- **Text to payload:** Copia el valor de la variable *filename* que se recibe mediante GET y lo almacena en la variable *msg.filename.*
- **File:** Nodo de entrada de fichero. Carga el fichero cuyo nombre se almacena en la variable *msg.filename* y pasa el dato contenido en ese fichero al siguiente nodo. Este nodo únicamente se encuentra disponible si se trabaja con una instancia local de Node-Red.
- **Speech to Text:** Nodo del servicio Speech to Text, se configura utilizando las credenciales del servicio creado.
- **Tanscription to payload:** Guarda en la variable msg.payload la transcripción del archivo de audio devuelto por el servicio STT.
- **HTTP response:** Devuelve como respuesta a la invocación un archivo de texto.

Flujo de interfaz

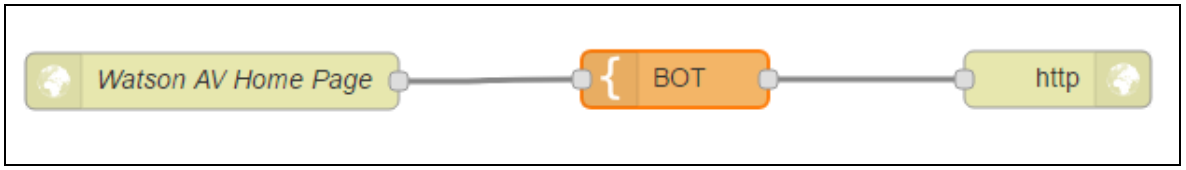


Figura 4-14: Flujo REST BOT.

- **Watson AV Home Page:** Nodo HTTP de entrada que se configura con el método de acceso GET y la ruta **/watsontalk.**
- **BOT:** Nodo *template* que se configura con código HTML, CSS y JavaScript. Es el código fuente de la página web Watson AV que se explica en la siguiente subsección.

- **HTTP response:** Carga la página HTML del *template*.

En el Anexo III se encuentran capturas de pantalla de cómo está configurado cada nodo de estos flujos.

4.3 Módulo Interfaz

Este apartado más que explicar específicamente qué elementos HTML y CSS se han utilizado para crear y dar estilo a la página web de la aplicación, va a centrar la atención en las funciones JavaScript que se utilizan para gestionar la entrada del usuario y realizar la comunicación con el módulo Servicio.

Comunicación con el flujo Conversation

Cuando se recibe la entrada de un usuario, ya sea de manera escrita o mediante voz, la página utiliza una función que toma los datos de entrada (en este caso un texto) y realiza una invocación al flujo Conversation utilizando la función *ajax* de la librería JQuery. En el mensaje enviado al flujo REST de Conversation, además del texto de entrada del usuario también se añade el contexto de la conversación (si existe en ese momento), como se muestra en la Figura 4-15.

invokeAjax('POST', 'watsontalk/botchat', txt, processOK);

```
function invokeAjax(method, path, message, okResponse) {
  var contextdata = $('#id_contextdump').data('convContext');
  var ajaxData = {};
  ajaxData.msgdata = message;
  if (contextdata) {
    ajaxData.context = contextdata;
  }
  $.ajax({
    type: method,
    url: path,
    data: ajaxData,
    success: okResponse,
    error: processNotOK
  });
}
```

Figura 4-15: Invocación mediante AJAX al flujo Conversation

Si todo ha ido bien, la respuesta se procesa mediante el método *proccesOK* que muestra el texto devuelto en el cuadro de mensajería de la app mediante la función *chat* y a continuación, llama a la función *audio* (se estudiará a continuación); además guarda el contexto devuelto por el servicio en una sección oculta para poder cargarlo al realizar la siguiente llamada al servicio. Este proceso se ilustra en la Figura 4-16.

```
function processOK(response) {
  chat(BOT, response.botresponse.messageout.output.text);
  audio(response.botresponse.messageout.output.text);
  $('#id_contextdump').data('convContext', response.botresponse.messageout.context);
}
```

Figura 4-16: Función *proccessOK.*

Comunicación con el flujo Text to Speech

La respuesta recibida de la invocación al flujo Conversation, además de mostrarse en pantalla, también se procesa con la función *audio* que se encarga de hacer una llamada al flujo Rest del servicio Text to Speech antes visto, como se muestra en la Figura 4-17.

```
function audio(txt) {
  document.getElementById('id_audio').innerHTML =
    '<audio controls autoplay>
      <source src="watsontalk/sayit?text_to_say=' + txt + '" type="audio/wav"></audio>';
}
```

Figura 4-17: Función *audio.*

A diferencia de la función anterior, esta función no hace una llamada al servicio utilizando AJAX, sino que, al ser de tipo GET el flujo TTS, se llama directamente añadiendo a la ruta de acceso *watsontalk/sayit* la variable *text_to_say* con el texto que se desea transformar a voz. Esta llamada devuelve un archivo de tipo audio/wav que se reproduce automáticamente en cuanto esté disponible.

Comunicación con el servicio Speech to Text y uso de Web Speech Api

Para transcribir a texto un archivo de audio que el usuario indique se hace una invocación al flujo Speech to Text de Node-Red. Esta invocación se realiza mediante la función *ajax* de la librería JQuery, pasándole en la variable *filename* el nombre del archivo que el usuario desee cargar (Figura 4-18):

```
$.ajax({
  type: 'GET',
  url: 'watsontalk/getFile',
  data: {
    filename: txt
  },
  success: processedVoice,
  error: processNotOK
});
```

Firgura 4-18: Invocación al servicio STT.

La invocación a este flujo devuelve el texto que el servicio Speech to Text transcribe a partir del archivo de audio. Este texto se procesa mediante la función *processedVoice* que hace una llamada al método *invokeAjax('POST', 'watsontalk/botchat', txt, processOK);* el cual, como se vio, realiza la comunicación con el servicio Conversation.

En segundo lugar, para el reconocimiento directo de voz se utiliza la Api Web Speech que tiene una interfaz llama *SpeechRecognition* que permite capturar voz desde un micrófono y contrastar las palabras que se reconocen con una gramática en específico. Cuando las palabras concuerdan con el vocabulario, el servicio devuelve la voz reconocida en forma de texto.

Con el uso del constructor *SpeechRecognition()* se crea una instancia de esta interfaz y se utiliza en el código JavaScript de la interfaz. Para crear el vocabulario se utiliza el constructor *SpeechGrammarList()* también proporcionado por el API (Figura 419).

```
var recognition = new SpeechRecognition();
var speechRecognitionList = new SpeechGrammarList();
recognition.grammars = speechRecognitionList;
recognition.lang = 'es';
recognition.interimResults = false;
recognition.maxAlternatives = 1;
```

Figura 4-19: Intancia de SpeechRecognition

A la intancia de reconocimiento se le da el nombre de *recognition* y a ella se le añade una serie de propiedades:

- **Grammars**: Añade la gramática a la instancia de reconocimiento.
- **Lang**: Se especifica el idioma que se desea reconocer. En este caso el español 'es'. Incluir esta propiedad limita toda la lista de gramática solo al español y facilita el reconocimiento.
- **InterimResults**: Se pone a *false* para indicar que se devuelva la palabra o frase reconocida directamente, sin resultados intermedios.
- **MaxAlternatives**: Se pone a 1 ya que solo se desea obtener una alternativa del resultado.

Cuando el usuario indica que quiere comunicarse por voz, se hace una llamada a *recognition.start()* que habilita las funciones controladores del reconocimiento, como se muestra en la Figura 4-20.

```
recognition.onresult = function(event) {
  start_img.src = mic;
  var speechResult = event.results[0][0].transcript;
  processedVoice(speechResult);
}

recognition.onspeechend = function() {
  start_img.src = mic;
  recognition.stop();
}

recognition.onerror = function(event) {
  start_img.src = mic_disable;
  __log('Ha occurrido un error al reconocer la voz: ' + event.error);
}
```

Figura 4-20: Funciones controladoras del reconocimiento de voz.

La función *recognition.onresult* se ejecuta si la interfaz ha reconocido la voz con éxito. El texto reconocido se encuentra dentro de la propiedad *transcript* de la respuesta devuelta, y es que se pasa a la función *processedVoice* vista arriba.

La función *recognition.onspeechend* se ejecuta cuando el usuario deja de hablar por un instante de tiempo en el que se produce silencio y en ese caso se termina la grabación mediante una llamada a *recognition.stop().* Por último la función *recognition.onerror* se ejecuta si ha ocurrido algún error en el reconocimiento y lo que hace es mostrar el error en la pantalla de Log.

En el Anexo IV se encuentran imágenes del sitio web de la aplicación.

5 Integración, pruebas y resultados

La manera en la que se ha organizado y desarrollado el proyecto ha permitido que las pruebas sobres los módulos y servicios implicados se realicen conforme se han creado, asegurando de esta manera que al realizar la integración ninguno de ellos presente un fallo individual.

A continuación, se muestran los resultados obtenidos al realizar pruebas individuales sobre los servicios de *IBM Watson*, y los resultados de las pruebas del sistema en entorno real, es decir, pruebas efectuadas sobre usuarios que interactúan con el asistente virtual desarrollado.

5.1 Pruebas sobre servicios de IBM Watson

Una vez creados los servicios de Watson, se debe comprobar que cada uno cumple con la funcionalidad que se espera. El servicio conversacional creado con Watson Conversation se probó utilizando el propio entorno de desarrollo de Bluemix, mientras que el resultado de la llamada a los servicios se probó utilizando la herramienta Postman, un invocador de servicios mediante el que se simulan las posibles peticiones que un usuario solicita a un servicio.

5.1.1 Pruebas en el servicio conversacional

El entorno de desarrollo del servicio de Watson Conversation ofrece un **Panel de Prueba** que se utilizó para verificar el resultado devuelto por el diálogo y las intenciones y entidades reconocidas para la entrada de prueba. La Figura 5-1 muestra un ejemplo del resultado devuelto por el servicio al introducir en el panel el texto "hola watson quiero encender mis luces":

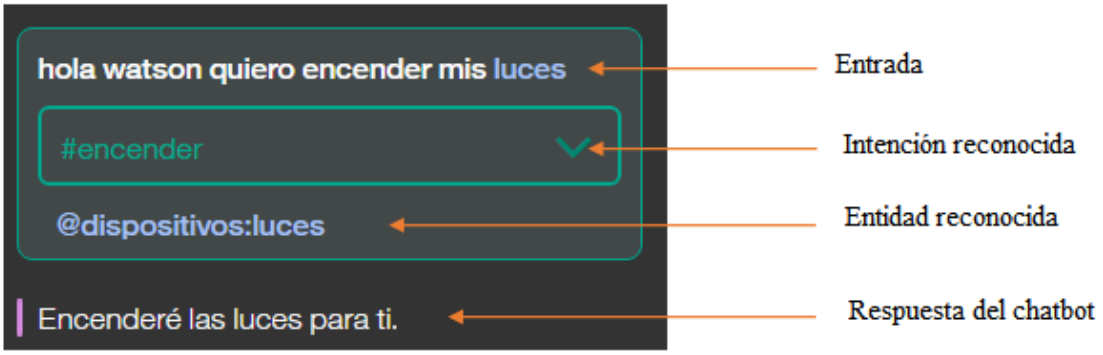


Figura 5-1: Resultado del Panel de Prueba

En el caso de que la intención reconocida no sea la adecuada para la entrada dada, se modifica el resultado obtenido manualmente, es decir, a través del Panel de Prueba se cambia la intención que reconoce el sistema por otra que se adecúe mejor a la entrada y contexto, como se muestra en la Figura 5-2.

Si para una entrada dada no se reconoce ninguna intención, mediante el Panel de Prueba se puede añadir esa entrada a una de las intenciones definidas o a una nueva. En el caso de que no se desee tomar en cuenta la entrada, se marca como "irrelevante", lo que indica al servicio que esa entrada no está relacionada con la aplicación. La Figura 5-2 muestra la opción de "Marcar como irrelevante" en el Panel de Prueba.

El entorno de desarrollo también permite ver el flujo seguido en el diálogo desde que se introduce un texto y reconoce la intención hasta que se devuelve la respuesta.

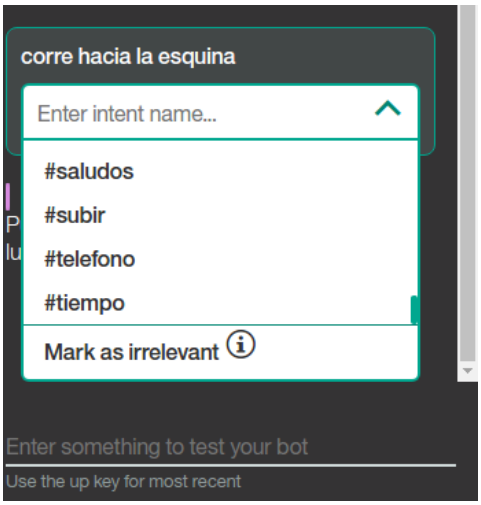


Figura 5-2: Modificación de intención en el Panel de Prueba

Las pruebas se fueron realizando conforme se creaban las intenciones, las entidades y el diálogo. Utilizando las funciones comentadas, se fue adecuando y mejorando el servicio conversacional para lograr que reconociera correctamente todas las intenciones/entidades y para que el flujo seguido en el diálogo devolviera una respuesta coherente y adecuada a la situación.

5.1.2 Pruebas de invocación

La herramienta invocadora de servicios Postman presenta una interfaz gráfica que permite realizar peticiones sobre servicios web de tipo REST y visualizar su resultado. Aunque la invocación a los servicios se hace mediante Postman, para observar mejor los parámetros, aquí se muestra en forma de comandos *curl* (herramienta de línea de comandos que permite obtener y enviar archivos utilizando sintaxis URL).

Para realizar las invocaciones, se utilizan las credenciales (URL, username y password) que se crean con los servicios (ver sección 4.2). A continuación, se muestran las pruebas de invocación realizadas.

Text to Speech

La llamada a este servicio se puede hacer de dos formas: utilizando el método HTTP POST o el método HTTP GET. A continuación, se muestran los comandos curl con los que se ha probado el servicio:

1. El comando de la Figura 5-3 utiliza el método HTTP POST para generar un archivo de audio de tipo WAV que contiene la voz con modelo es-ES_EnriqueVoice (castellano) que el servicio Text to Speech obtiene del texto “hola mundo” pasado como dato:

```
curl -X POST -u {username}:{password}
    --header "Content-Type: application/json"
    --header "Accept: audio/wav"
    --data "{\"text\":\"hola mundo\", \"voice\":\"es-ES_EnriqueVoice\"}"
    --output test.wav
    "https://stream.watsonplatform.net/text-to-speech/api/v1/synthesize"
```

Figura 5-3: Invocación POST al servicio Text to Speech

2. El segundo comando (Figura 5-4) utiliza el método HTTP GET para llamar al servicio que sintetiza el texto "hola mundo" con la misma voz que el comando anterior:

```
curl -X GET -u {username}:{password}
    --output hola_mundo.wav
    "https://stream.watsonplatform.net/text-to-speech/api/v1/synthesize?
    accept=audio/wav&text=hola%20mundo&voice=es-ES_EnriqueVoice"
```

Figura 5-4: Invocación GET al servicio Text to Speech

Ambos comandos hacen una llamada a la dirección HTTP del servicio Text to Speech "https://stream.watsonplatform.net/text-to-speech/api/v1/synthesize" que se encarga de sintetizar el texto. Ambos métodos responden con el archivo de audio esperado.

Speech to Text

Al igual que con Text to Speech, se prueba el funcionamiento de este servicio utilizando un comando curl. Para la prueba se utiliza un archivo de nombre *test.wav* que contiene el audio "hola cómo estás".

El comando de la Figura 5-5 envía el archivo de audio al servicio utilizando el modelo es-ES_BroadbandModel para obtener la transcripción del audio en castellano.

```
curl -X POST -u {username}:{password}
--header "Content-Type: audio/wav"
--data-binary @test.wav
"https://stream.watsonplatform.net/speech-to-text/api/v1/recognize?model=es-ES_BroadbandModel"
```

Figura 5-5: Invocación POST al servicio Speech to Text

El resultado obtenido viene en formato JSON y contiene información sobre la transcripción, como se muestra en la Figura 5-6:


```
{
  "results": [
    {
      "alternatives": [
        {
            "confidence": 0.826,
            "transcript": "hola cómo estás "
        }
      ],
      "final": true
    }
  ],
  "result_index": 0
}
```


Figura 5-6: Resultado de invocación al servicio Text to Speech

El resultado muestra un array con las posibles alternativas de la transcripción del audio, en este caso únicamente una, con una confianza (confidence) de transcripción de un 82.6%. La etiqueta "final: true" indica que es el resultado final y la etiqueta "result_index: 0" indica que es el primer (y único) resultado del grupo de respuesta.

Watson Conversation

Para probar que este servicio es capaz de procesar correctamente un mensaje y devolver una respuesta del diálogo se utiliza el comando curl de la Figura 5-7 que envía el mensaje "Enciende las luces" al Watson Conversation.

```
curl -X POST -u {username}:{password}
--header "Content-Type:application/json"
--data "{
          \"input\": {\"text\": \"Enciende las luces\"}
        }"
"https://gateway.watsonplatform.net/conversation/api/workspaces/44d94bb6-49c9-42b9-9cd8-d07d2b2e3a63/message/"
```

Figura 5-7: Invocación POST al servicio Watson Conversation

La Figura 5-8 muestra el resultado devuelto, se ve cómo el servicio reconoce la intención *#encender* y la entidad *@dispositivos* con una confianza del 100%. El servicio muestra también el texto de salida "Encenderé las luces para ti.", los nodos que se han visitado y la acción efectuada "lights_on".

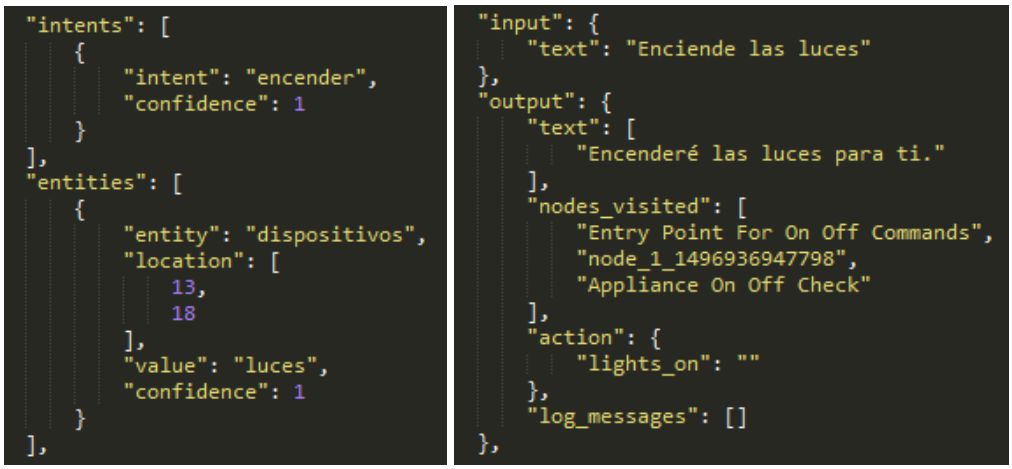


Figura 5-8: Intenciones y entidades reconocidas / Entrada y salida del servicio

5.2 Pruebas en entorno real

Una vez se han integrado todos los módulos con flujos de Node-Red, y la aplicación se encuentra desplegada en un servidor (local o de Bluemix), el sistema se encuentra disponible para su uso en un entorno real.

Para probar si el sistema desarrollado cumple con las características propias de un agente virtual (ver sección 2.1), se ha elegido a un grupo de 12 personas que interactúan con el sistema a través de la interfaz creada, haciéndole peticiones, preguntas y tratando de mantener una conversación (escrita y oral) con el agente virtual. Los resultados de las pruebas se han recogido de dos formas:

- La primera se hace a través del propio entorno de desarrollo de Watson Conversation que incluye un componente que permite visualizar todas las conversaciones que los usuarios han tenido con el chatbot. De esta forma, se tiene una visión global de aquellas peticiones (intenciones + entidades) más comunes en los usuarios, permitiendo mejorar la manera en la que está organizado el diálogo y las respuestas que proporciona.
- La segunda forma es a través de un formulario que se presenta a los usuarios al terminar la prueba con el asistente virtual y cuyo cometido es evaluar la

satisfacción de los usuarios con respecto a la interacción que han tenido con el sistema. Las preguntas utilizadas en el formulario se encuentran disponible en el Anexo V.

5.2.1 Resultados

Mediante el entorno de desarrollo de Watson Conversation, analizando las conversaciones realizadas por los usuarios, se obtienen las estadísticas mostradas en la Figura 5-9.

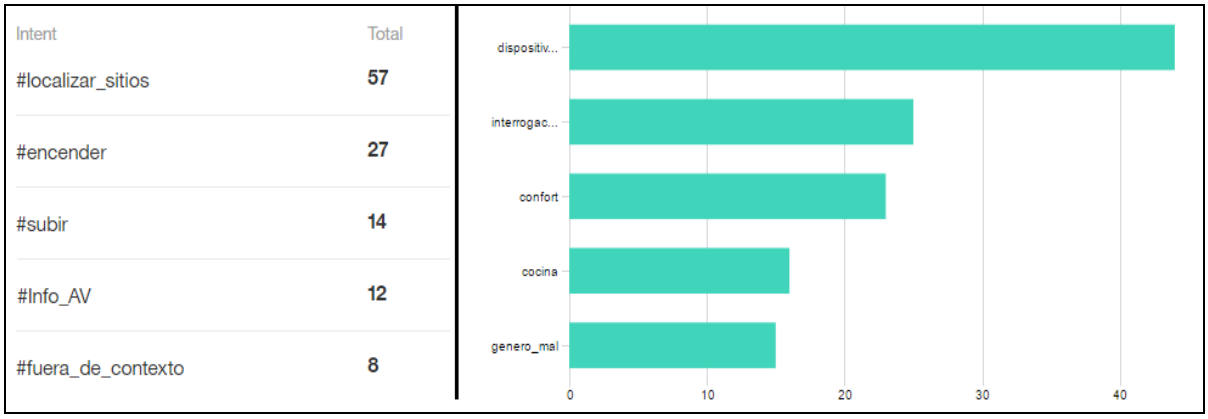


Figura 5-9: Top 5 Intenciones / Top 5 Entidades

La figura anterior muestra el "top 5" de intenciones y entidades reconocidas por el servicio conversacional. Para mejorar el diálogo y el entendimiento del lenguaje natural (reconocimiento de intenciones y entidades), teniendo en cuenta estos datos, se podría "entrenar" a las intenciones y entidades más populares añadiendo más variedad de ejemplos o modificando los ya existentes.

De la encuesta realizada a los usuarios que probaron Watson AV se obtiene los resultados que se muestran en la Figura 5-10.

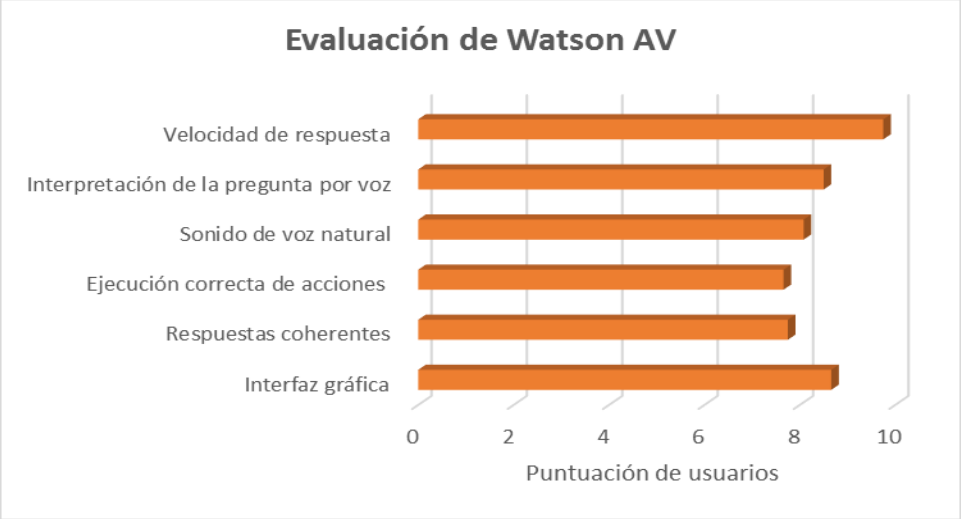


Figura 5-10: Resultado de evaluación de Watson AV

La figura anterior muestra una media de las puntuaciones de las 6 preguntas evaluadas por los usuarios. En general, el asistente virtual obtiene una puntuación en conjunto de aproximadamente 9/10, puntuación que se puede mejorar realizando las modificaciones propuestas por los usuarios.

En el Anexo VI se encuentran los detalles de cada respuesta.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En este proyecto se ha desarrollado y presentado un agente virtual con capacidades cognitivas utilizando los servicios de IBM Watson, accesible a través de un servidor web y diseñado para interactuar de manera natural con personas y realizar acciones sobre el panel de control de un automóvil.

Durante el proceso de desarrollo ha sido fundamental la detección de las necesidades de los clientes que utilicen este servicio, para cumplir con el objetivo principal, elaborar un sistema capaz de entender y procesar el lenguaje natural, interactuar de manera escrita u oral y aprender y razonar para dar una respuesta concisa y adecuada a la situación en la que se encuentre, además de realizar las acciones que se le pida.

Al término de este Trabajo de Fin de Grado, una de las conclusiones derivadas de la experiencia con usuarios es que una herramienta como un asistente virtual presenta una gran utilidad en el entorno empresarial dedicado al comercio electrónico o a la prestación de servicios por los bajos costes que supone implantar el sistema y por los beneficios que proporciona tanto a los clientes como al negocio. Además, el auge cada día de nuevas tecnologías y el llamado *Internet de las Cosas* hacen cada vez más necesario el uso de aplicaciones "humanizadas" que permitan a los usuarios realizar acciones a través de sus dispositivos de una forma rápida, sencilla e intuitiva.

Como conclusiones finales de este Trabajo de Fin de Grado se tienen:

- El desarrollo de un sistema conversacional que facilita la comunicación e interacción entre los clientes y el servicio que se proporciona.
- Un sistema que proporciona recomendaciones individualizadas mediante la comprensión de la personalidad del usuario y sus gustos.
- Un agente virtual que posee una capacidad de comunicación escrita y oral, mejorando la satisfacción general del cliente.
- Aunque el bot conversacional se ha desarrollado de manera específica para prestar un determinado servicio, puede servir como base para el diseño de asistentes virtuales enfocados a distintas áreas, debido a la flexibilidad que presenta el sistema.
- Una aplicación compuesta por elementos que se han priorizado para ser fácilmente reutilizables y extensibles a nuevas tecnologías y entornos de despliegue.

6.2 Trabajo futuro

El agente virtual desarrollado admite varias mejoras, tanto en el diseño como en la funcionalidad que presenta. Varias de estas mejoras han surgido como resultado de las pruebas realizadas en usuarios, ya que permitieron ver cuáles son las verdaderas acciones e intenciones que una persona común (ajena al proyecto) realiza y necesita con más frecuencia.

Como líneas de trabajo futuro se propone para el proyecto las siguientes:

- Ampliar la funcionalidad del asistente virtual permitiendo a los usuarios introducir imágenes en la aplicación que el servicio sea capaz de procesar y, a partir de ellas, realizar la recomendación de un servicio. Este reconocimiento de imagen se podría implementar utilizando el servicio de **Visual Recognition** disponible en la plataforma de IBM Bluemix y perteneciente al conjunto de Watson Services.

- Ampliar la funcionalidad del sistema permitiendo que los usuarios se comuniquen con el servicio en varios idiomas.

- Ampliar la funcionalidad del bot conversacional para permitir, además de recomendar servicios y realizar acciones, mantener un diálogo coherente sobre cualquier temática con el usuario.

- Enriquecer la aplicación añadiendo información de redes sociales. Esto se conseguiría solicitando a los usuarios su cuenta en alguna red social e invocando a servicios Watson como Natural Language Understanding para interpretar su personalidad.

- Añadir otra interfaz de diálogo para que, además de ser en navegador, sea una aplicación en Smartphone u otra interfaz como Twilio, una API de servicio web de telefonía que le permite utilizar las habilidades y lenguajes web existentes para crear aplicaciones de voz (llamadas) y SMS.

- Ampliar el conjunto de capacidades actuales del agente virtual para satisfacer las necesidades de los usuarios.

- Añadir más respuestas y mejorar las ya existentes para que el bot conversacional brinde una sensación de comunicación más cercana a la humana.

Referencias

[1] *Listen to Customer's Demands for Better Service*. Available: http://www.gartner.com/technology/home.jsp.

[2] En 2020, más de 30 mil millones de dispositivos estarán conectados a internet | OBS business school. | OBS business school. Available: http://www.obs-edu.com/es/noticias/estudio-obs/en-2020-mas-de-30-mil-millones-de-dispositivos-estaran-conectados-internet.

[3] M. Jabois. M. Jabois. "En 2020 hablaremos más con un bot que con nuestro novio". "En 2020 hablaremos más con un bot que con nuestro novio". *El País El País* 2017. Available: http://tecnologia.elpais.com/tecnologia/2017/06/03/actualidad/1496490877_972265.html.

[4] *e-commerce - ¿Qué es el e-commerce?*. Available: https://debitoor.es/glosario/definicion-ecommerce.

[5] D. Wellers, T. Elliott and M. Noga. (-05-31T14:00:44Z). *8 Ways Machine Learning Is Improving Companies' Work Processes*. Available: https://hbr.org/2017/05/8-ways-machine-learning-is-improving-companies-work-processes.

[6] Test de turing. Test de turing. 2017. Available: https://es.wikipedia.org/w/index.php?title=Test_de_Turing&oldid=99676341.

[7] Bot conversacional. Bot conversacional. 2017. Available: https://es.wikipedia.org/w/index.php?title=Bot_conversacional&oldid=99565255.

[8] Jabberwacky. Jabberwacky. 2016. Available: https://en.wikipedia.org/w/index.php?title=Jabberwacky&oldid=699987131.

[9] Artificial linguistic internet computer entity. Artificial linguistic internet computer entity. 2017. Available: https://en.wikipedia.org/w/index.php?title=Artificial_Linguistic_Internet_Computer_Entity&oldid=772651671.

[10] C. Salza. (). *Los asistentes virtuales, los nuevos copilotos de los coches conectados*. Available: http://prnoticias.com/tecnologia/innovacion/20159695-asistentes-virtuales-coches-conectados.

[11] Bengio, Y, 2013, Representation Learning: A Review and New Perspectives.

[12] *Watson Virtual Agent*. Available: https://www.ibm.com/us-en/marketplace/cognitive-customer-engagement.

[13] *IBM Bluemix - Plataforma de desarrollo de apps en la nube de próxima generación*. Available: https://console.ng.bluemix.net/.

[14] (Sep 7,). *Speech Synthesis Markup Language (SSML) Version 1.0*. Available: https://www.w3.org/TR/speech-synthesis/.

[15] *Node-Red*. Available: https://nodered.org/#features.

[16] *Bootstrap · The world's most popular mobile-first and responsive front-end framework.* Available: http://getbootstrap.com/.

[17] E. Castledine and C. Sharkie. E. Castledine and C. Sharkie. "What's so good about jQuery?" in *jQuery: Novice to Ninja* (2nd ed.) (2nd ed.)Anonymous 2012, Available: http://proquest.safaribooksonline.com/book/programming/javascript/9780987153012/falling-in-love-with-jquery/i_section1_d1e490.

[18] Sankaranarayanan, Sreelantha, 2016, Learning IBM Bluemix.

[19] *Conversation | System entities | IBM Watson Developer Cloud*. Available: https://www.ibm.com/watson/developercloud/doc/conversation/system-entities.html.

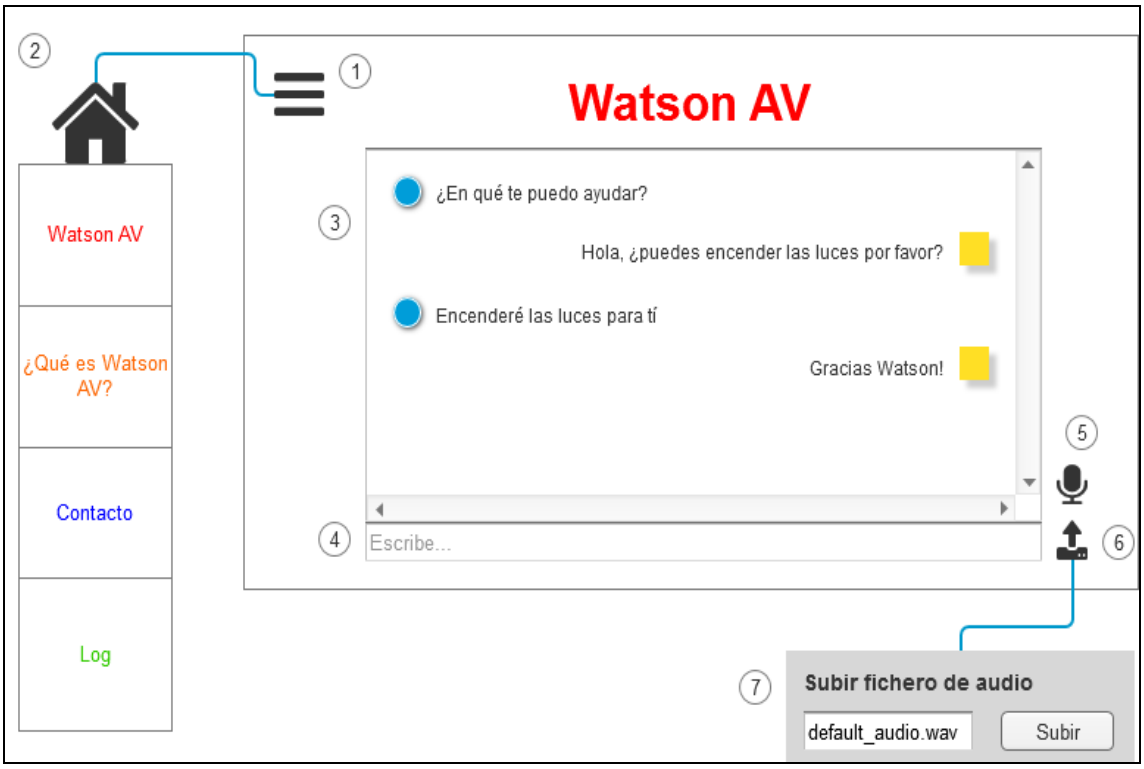
Glosario

AJAX	Asynchronous JavaScript and XML
API	Interfaz de Programación de Aplicaciones (Application Programming Interface)
AV	Asistente Virtual
CSV	Comma-Separated Value
IA	Inteligencia Artificial
IoT	Internet of Things
IPA	International Phonetic Alphabet
JSON	JavaScript Object Notation
PaaS	Platform as a Service
RRSS	Redes Sociales
SaaS	Software as a Service
SPR	Symbolic Phonetic Representation
SSML	Speech Synthesis Markup Language
STT	Speech to Text
TTS	Text to Speech

Anexos

I. Maquetas de la aplicación Web

Watson AV



Leyenda:

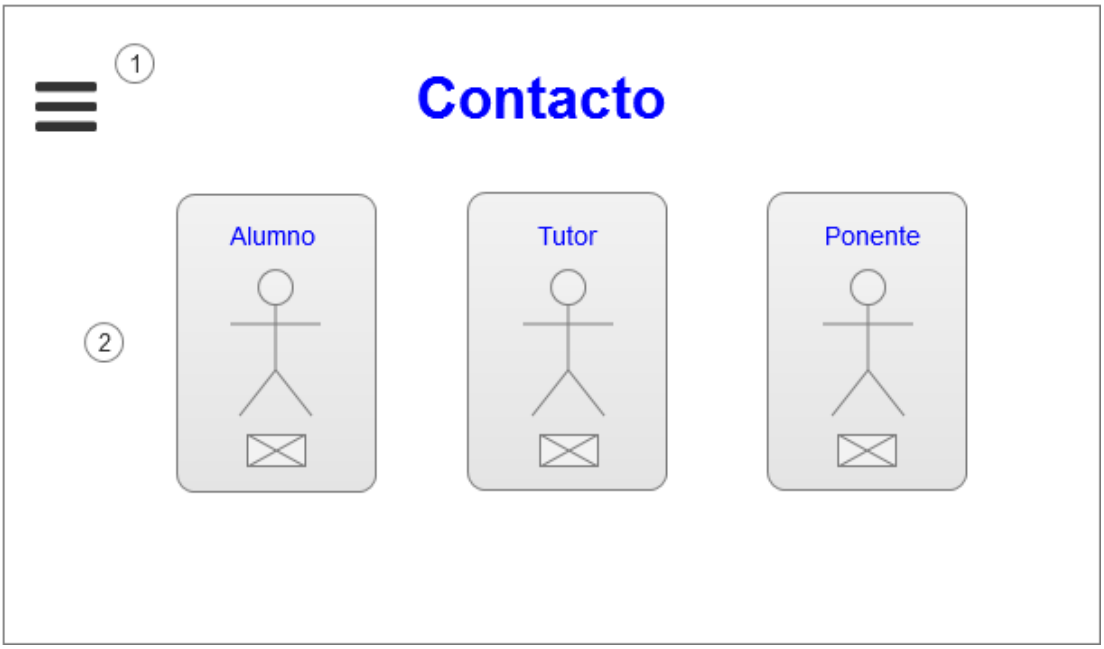
1. Pestaña para desplegar el menú lateral.
2. Menú lateral con los enlaces a las páginas de aplicación.
3. Área de mensajería.
4. Cuadro para escribir un mensaje al asistente virtual.
5. Botón micrófono. Al accionarlo empieza a grabar audio.
6. Botón para subir un archivo de audio. Al accionarlo abre un cuadro de diálogo para seleccionar un fichero.
7. Cuadro de diálogo para seleccionar un fichero.

¿Qué es Watson AV?

Leyenda:

1. Pestaña para desplegar el menú lateral.
2. Área que contiene la información del asisten virtual.

Contacto



Leyenda:

1. Pestaña para desplegar el menú lateral.
2. Área con información de contacto.

Log

Leyenda:

1. Pestaña para desplegar el menú lateral.
2. Área donde se muestra la información.

II. Intenciones y Entidades

Intenciones

En el proyecto se definen 25 grupos de intenciones distintas que se reflejan en la siguiente tabla junto con una pequeña descripción y su ámbito.

Intención	Descripción	Ámbito
#actualizaciones_trafico	Agrupa aquellas cuestiones referentes a la situación del tráfico. Ej.: *¿Hay atascos?*	Específica
#apagar	Reúne las acciones que representan apagar o desactivar "algo". Ej.: *Desactivar radio.*	Específica
#bajar	Agrupa las acciones que representan reducir "algo". Ej.: *Bajar el volumen de la música.*	Específica
#capacidades	Agrupa cuestiones relativas a las acciones que puede hacer el AV. Ej.: *¿De qué eres capaz?*	General
#confianza_sistema	Reúne aquellas sentencias que cuestionan la confiabilidad del sistema. Ej.: *¿Me mientes?*	General
#despedidas	Asocia todas las frases referentes a la acción de despedirse. Ej.: *Adiós. Hasta mañana.*	General
#encender	Asocia todas las acciones referentes a encender o activar "algo". Ej.: *Enciende la calefacción.*	Específica
#errores_interfaz	Reúne las acciones que representan errores en el funcionamiento del sistema: Ej.: *No hay sonido.*	Específica
#fuera_capacidades	Aquí se agrupan las acciones que exceden la funcionalidad del sistema. Ej.: *Cambiar marcha.*	Específica
#fuera_de_contexto	Agrupa acciones, preguntas o frases que se encuentran fuera de contexto. Ej.: *¿Me amas?*	General
#info_AV	Reúne preguntas referentes al asistente virtual. Ej.: *¿Cuál es tu música favorita?*	Específica

#interacciones_interfaz	Agrupa preguntas o frases sobre la manera en la que se interactúa con el AV. Ej.: *¿Puedes oírme?*	General
#localizar_sitios	Agrupa preguntas o acciones que indica al AV que busque un lugar. Ej.: *¿Dónde hay comida?*	Específica
#mejoras_sistema	Asocia entradas del usuario que intentan actuar sobre el sistema. Ej.: *¿Puedo enseñarte?*	General
#navegación	Agrupa preguntas referentes al viaje. Ej.: *¿Cuánto tiempo lleva ir a una gasolinera desde aquí?*	Específica
#no_especificado	Aquí se incluyen sentencias que indican que el AV puede tomar cualquier decisión. Ej.: *Elije tú*	General
#peticiones_compuestas	Agrupa frases que incluyen más de una petición o acción. Ej.: *Apaga y enciende la luz.*	General
#presentación	Aquí se agrupan las frases que se entienden como presentación del usuario. Ej.: *Mi nombre es…*	General
#reacciones_negativas	Relaciona entradas del usuario catalogadas como negativas. Ej.: *No sabes hacer nada!*	General
#reacciones_positivas	Relaciona entradas del usuario catalogadas como positivas o de agradecimiento. Ej.: *Mil gracias!*	General
#respuesta_y_decisiones	Reúne entradas que pueden ser repuestas por parte del usuario. Ej.: *Sí, por favor*.	General
#saludos	Asocia todas las frases referentes a la acción de despedirse. Ej.: *Buenas, ¿qué tal?*	General
#subir	Agrupa las acciones que representan incrementar “algo”. Ej.: *Subir la calefacción.*	Específica
#teléfono	Agrupa acciones relativas a funciones telefónicas.	Específica

	Ej.: *Puedes llamar al 112*	
#tiempo	Agrupa cuestiones relativas al clima. Ej.: *¿Cuándo saldrá el sol?*	General

Categorías de intenciones

*Aclaración: Los nombres de las categorías se han escrito tal cual se utilizan en el entorno de desarrollo. Se obliga a que todas las intenciones empiecen por el símbolo **#.***

Entidades

Teniendo en cuenta las intenciones que el asistente virtual del proyecto es capaz de reconocer, se definen 14 grupos de entidades recogidos en la siguiente tabla junto con una breve descripción, un ejemplo de los valores de cada grupo y la intención/intenciones en la que se aplica:

Entidad	Descripción	Intención
@cocina	Aquí se encuentran los tipos de comida que el AV es capaz de recomendar. Ej.: *pasta, tacos, hamburguesa…*	#localizar sitios #info_AV
@cocina_mal	Al contrario que *@cocina*, aquí se encuentran los tipos de comida que el AV no puede recomendar. Ej.: *israelí, kosher…*	#localizar sitios #info_AV
@condicion_clima	Aquí se encuentran varias condiciones que se pueden dar sobre el clima. Ej.: *frío, calor, soleado…*	#tiempo
@confort	En esta entidad se hallan los lugares de ocio que el asistente puede recomendar. Ej.: *restaurante, cine…*	#info_AV #localizar_sitios #navegacion
@dispositivos	Aquí se localizan los dispositivos que asistente puede controlar dentro del vehículo. Ej.: *ventilador, música, luces…*	#apagar, #encender, #bajar, #subir
@genero	Géneros musicales que el agente virtual es capaz de reproducir. Ej.: *pop, rock, jazz…*	#encender
@genero_mal	Al contrario que *@generos*, aquí se encuentran aquellos tipos de música que el AV no es capaz de reproducir. Ej.: *country, dance...*	#encender
@informacion_av	En esta entidad se incluyen términos que	#Info_AV

	representan información del asistente virtual. Ej.: *nombre, nacimiento…*	
@interrogación	Aquí se incluyen aquellas palabras que puedan significar el inicio de una interrogación. Ej.: *Quién, cuál, cuándo…*	#Info_AV, #interacciones_inter., #localizar_sitios, #capacidades
@nombres	Aquí se incluyen varios nombres propios de personas que el AV puede reconocer. Ej.: *Pedro, Juan …*	#presentacion
@opcion	Aquí se encuentran tipos de opciones que el usuario puede elegir y el AV reconocer. Ej.: *tercero, quinto, último…*	#localizar_sitios
@START	Entidad especial que solo almacena un valor, *conversation_start.* Se utiliza como entrada para iniciar la conversación (es transparente al usuario)	Inicio de conversación
@telefono	Agrupa las opciones que tiene un teléfono. Ej.: *llamada, sms…*	#telefono
@tipos_de_respuesta	Aquí se encuentran los tipos de respuesta del usuario. Ej.: *negativa, positiva, incierta*	Depende del contexto

Categorías de entidades

Aclaración: Los nombres de las categorías se han escrito tal cual se utilizan en el entorno de desarrollo. Se obliga a que todas las entidades empiecen por el símbolo @.

A parte de las entidades propias que se pueden definir, el servicio ofrece cinco categorías más (+2 en BETA, disponible sólo para inglés) llamadas **entidades del sistema** que al ser activadas se reconocen automáticamente en cualquier situación en la que intervengan [19]:

1. **@sys-time:** Reconoce menciones de tiempo () en varios formatos y guarda la referencia como una cadena de texto (string).
   Formatos: 17h15, 11:30, ahora, 5 pm, 14:10
2. **@sys-date:** Reconoce menciones de fechas en varios formatos y guarda una referencia a ellas como una cadena de texto.
   Formatos: viernes, hoy, 23 de agosto de 2017, ahora
3. **@sys-currency:** Reconoce y extrae el valor de una cantidad de dinero expresada en varios formatos. Guarda la cantidad en una cadena de texto.
   Formatos: 50 céntimos, $10, 50.20€
4. **@sys-percentage:** Reconoce un porcentaje introducido por el usuario y extrae su valor guardado una referencia a él en formato de texto.
   Formatos: 25%, 88 porciento
5. **@sys-number:** Extrae el valor numérico de una entrada del usuario y guarda una referencia como cadena de texto representando el valor.

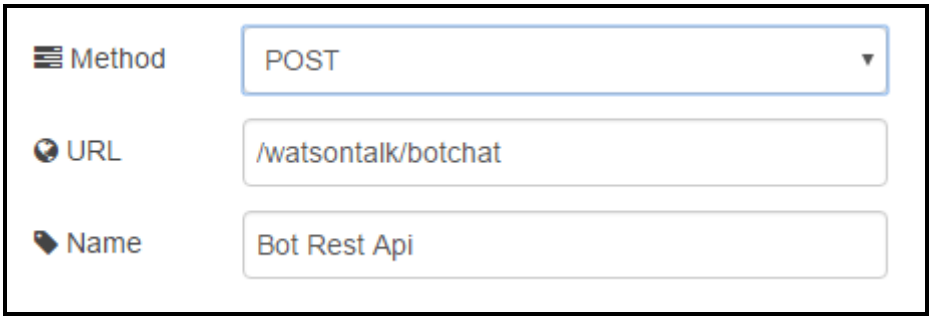
Formato: 58, treinta y dos, π, 3.1416

6. **@sys-location [En BETA]:** Reconoce nombres de lugares como países, estados, provincias, pueblos, etc.
   Formato: New York, U.S.A
7. **@sys-person [En BETA]:** Reconoce nombres propios de personas, simples o compuestos. Cuando esté completamente desarrollada, esta entidad se podría utilizar como sustituta a la entidad @nombres utilizada en el proyecto.
   Formato: Jhon, Jhon Smith.

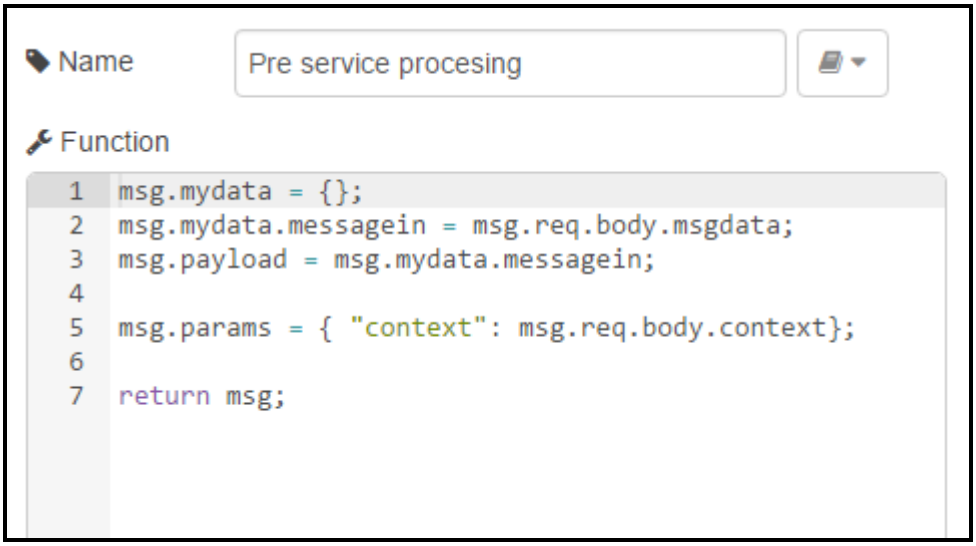
III. Configuración de flujos Node-Red

A continuación, se va a mostrar cómo están configurados los nodos de los flujos creados en la aplicación Node-Red.

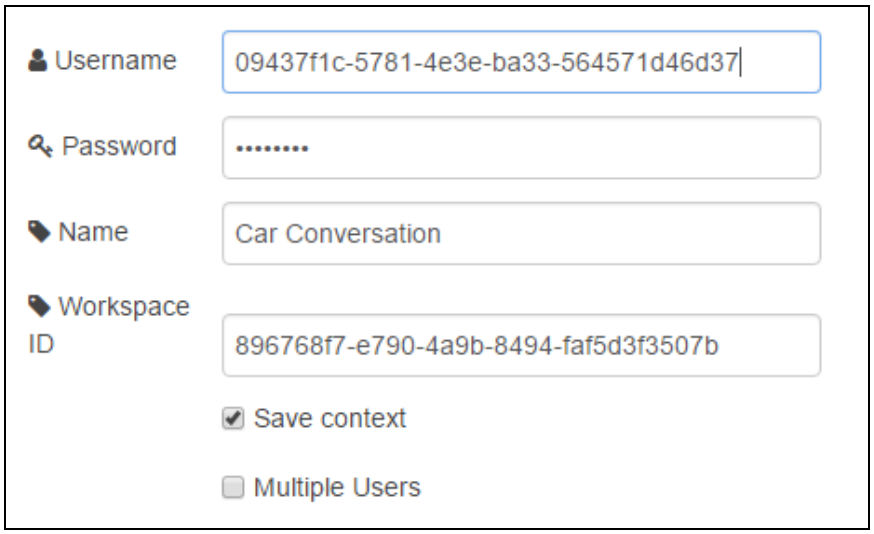
Flujo Conversation



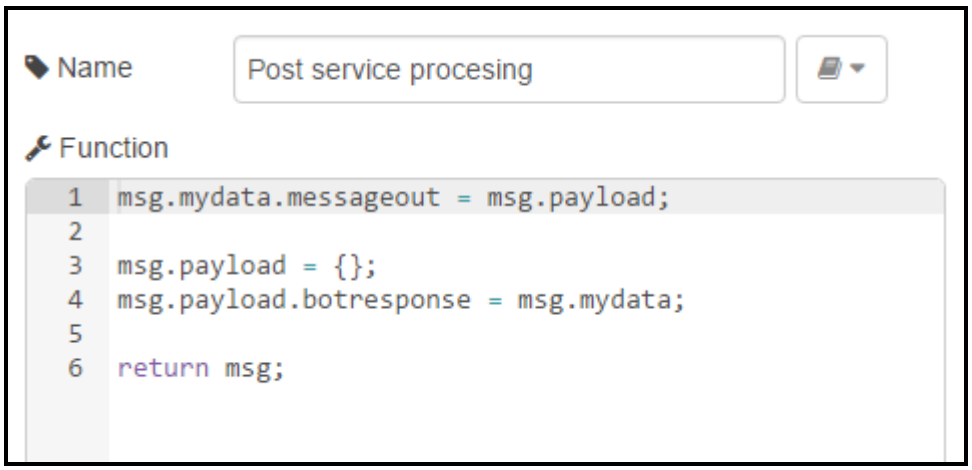
Nodo "Bot Rest Api"



Nodo "Pre service procesing"

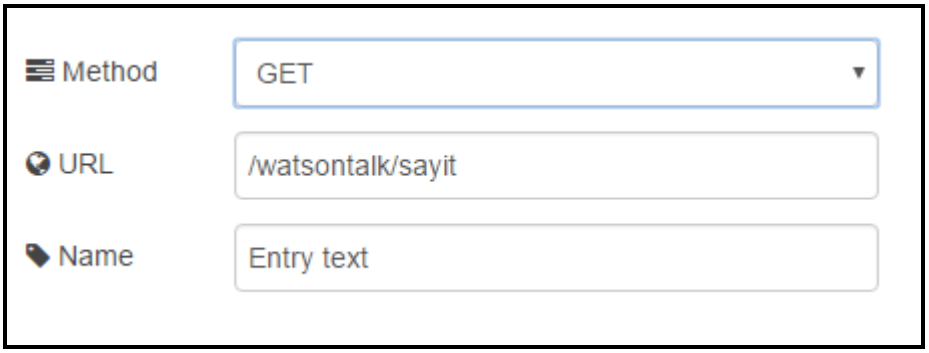


Nodo "Car Conversation"

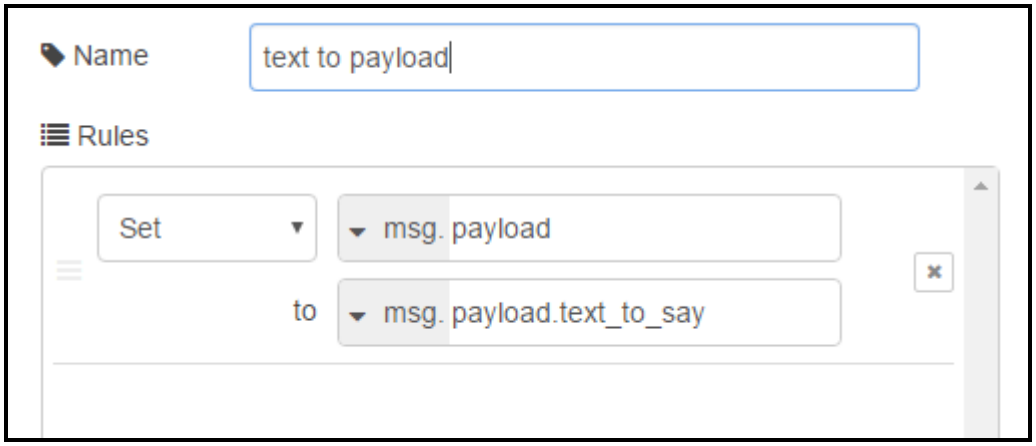


Nodo "Post service procesing"

Flujo Text to Speech



Nodo "Entry text"

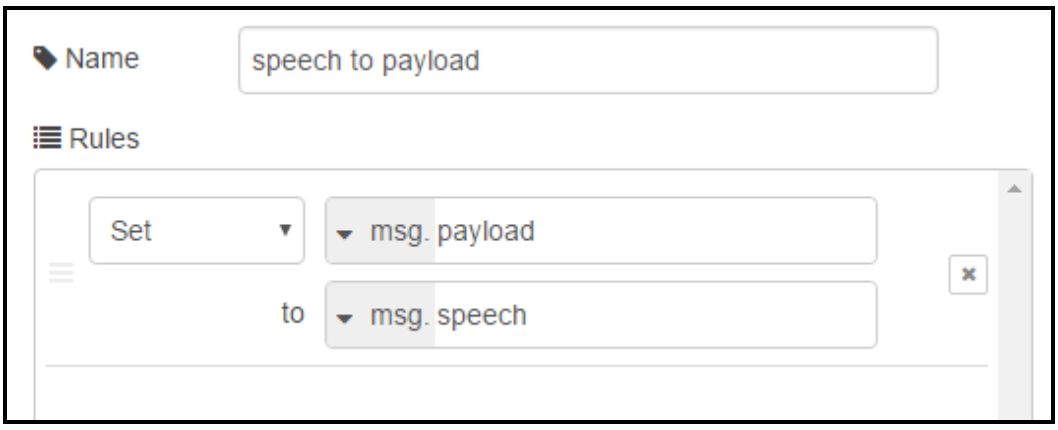


Nodo "Text to payload"

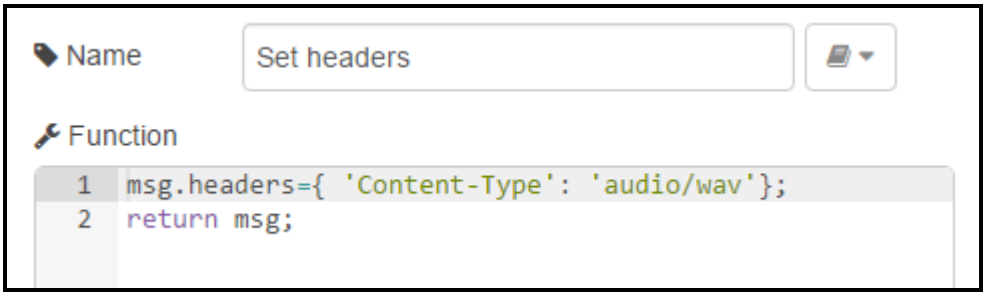
Nodo "Add SPR's"

Nodo "Add Expressiveness"

Nodo "TTS"

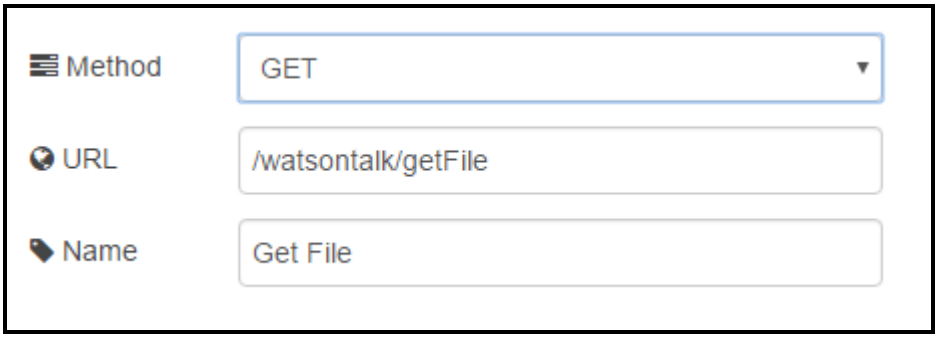


Nodo "Speech to payload"



Nodo "Set headers"

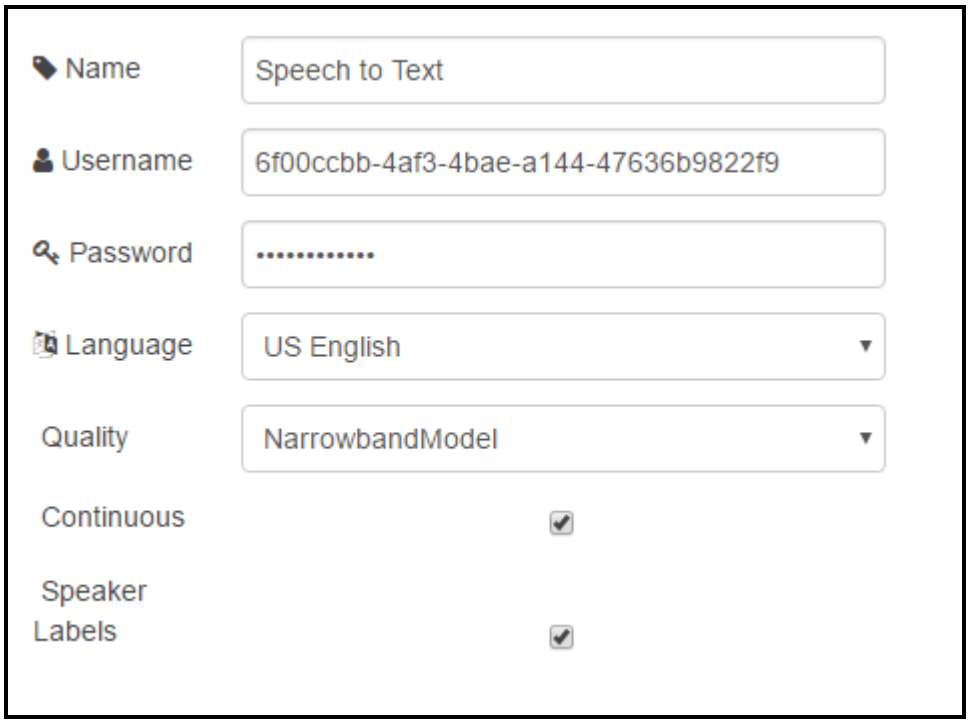
Flujo Speech to Text



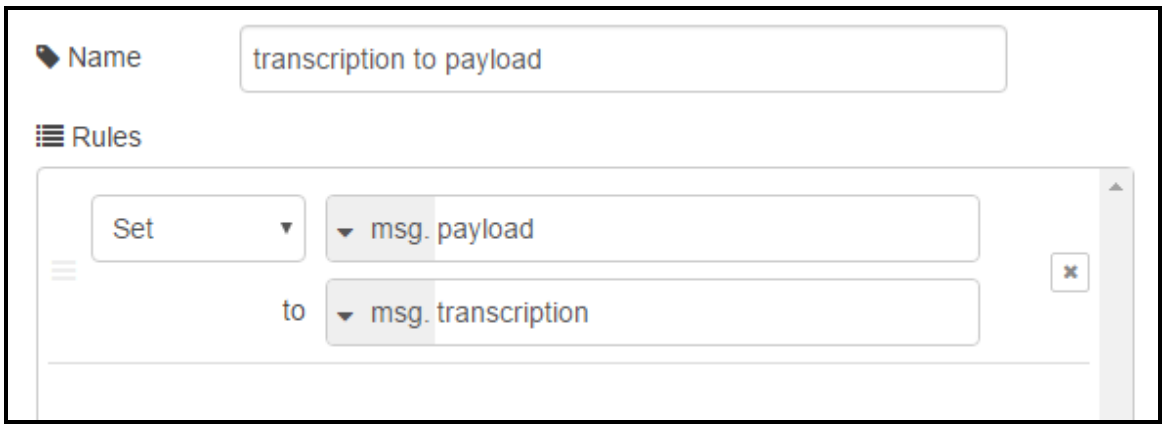
Nodo "Get file"

Nodo "Text to filename"

Nodo "File"

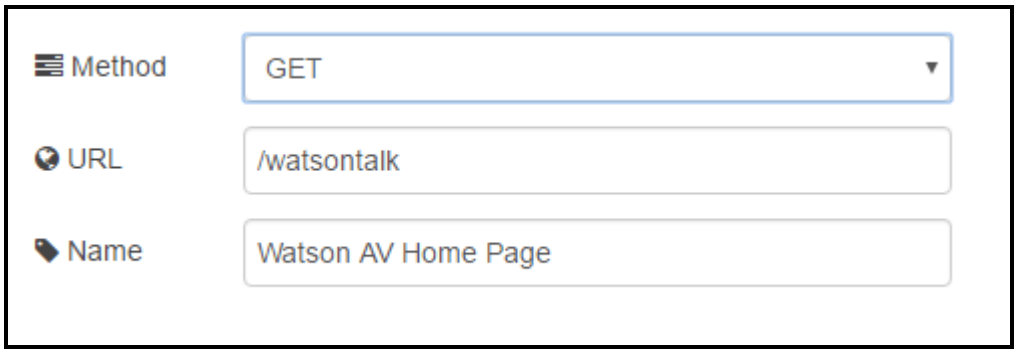


Nodo "Speech to Text"

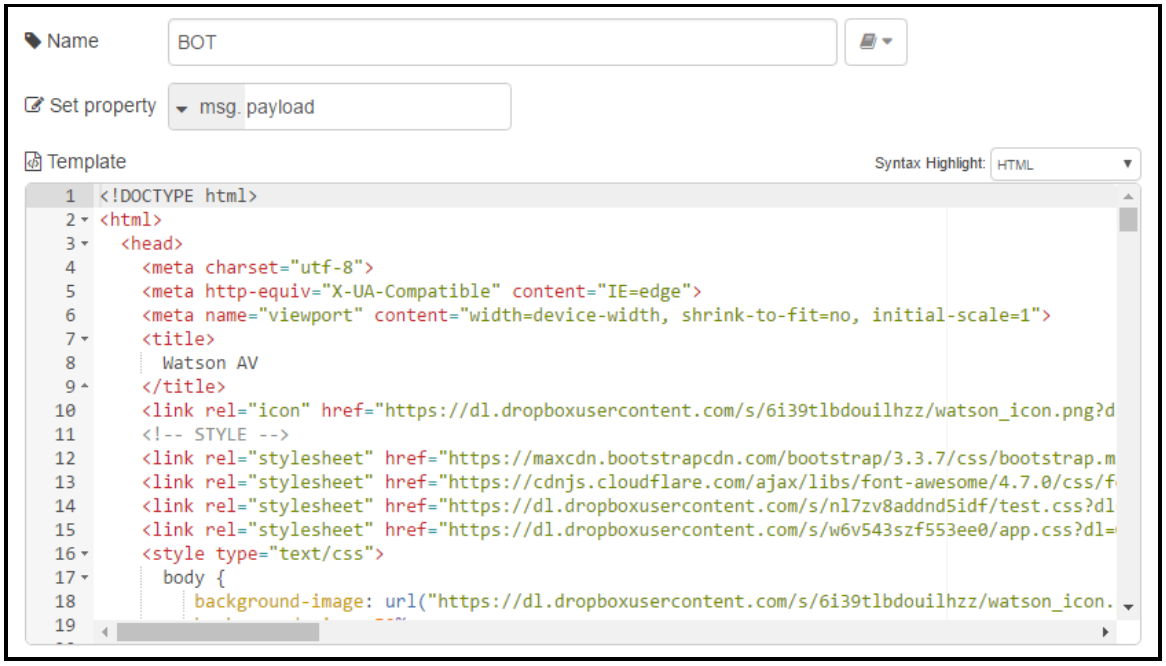


Nodo "Transcription to Payload"

Flujo Watson AV



Nodo "Watson AV Home Page"

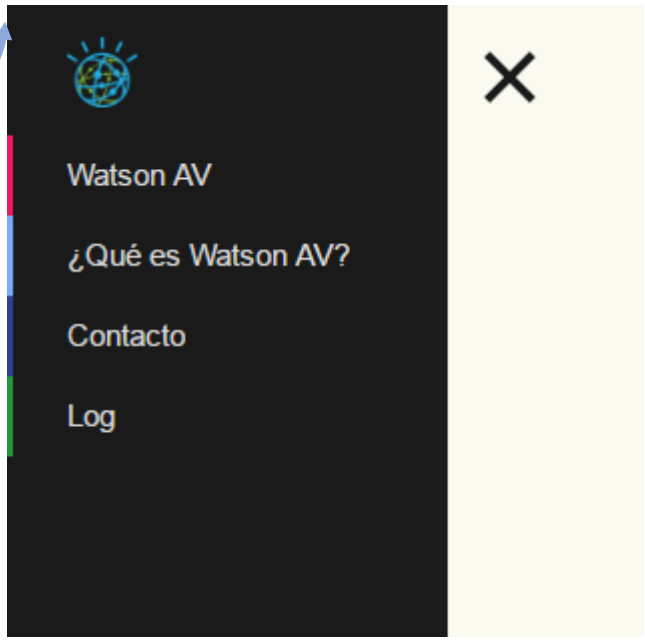


Nodo "BOT"

Nota: Este último no contiene el código HTML de la Interfaz. Aquí solo se muestra la parte inicial del código.

IV. Sitio Web de Watson AV

Watson AV



¿Qué es Watson AV?

Contacto

Log


Log
Procesando...
Texto procesado correctamente.
Escuchando...
Grabación detenida
Procesando...
Texto procesado correctamente.

V. Formulario de evaluación

El formulario de evaluación cuenta con dos secciones: Evaluación de Watson AV y Mejoras sobre Watson AV:

Evaluación de Watson AV

En esta sección se debe contestar una serie de preguntas sobre la interacción realizada con el asistente virtual, puntuando en una escala de 1 a 10.

1. ¿En qué grado el asistente virtual ha respondido coherentemente a las preguntas y peticiones realizas en la conversación?
2. ¿En qué grado el asistente virtual ha cumplido correctamente con las acciones que se le han pedido?
3. ¿En qué grado la voz del asistente virtual se escucha de manera natural?
4. ¿En qué grado el asistente virtual ha interpretado bien la pregunta o petición realizada mediante voz?
5. ¿Cuál ha sido el grado de velocidad en las respuestas del asistente virtual?
6. ¿En qué grado el diseño de página web resulta amigable?

Mejoras sobre Watson AV

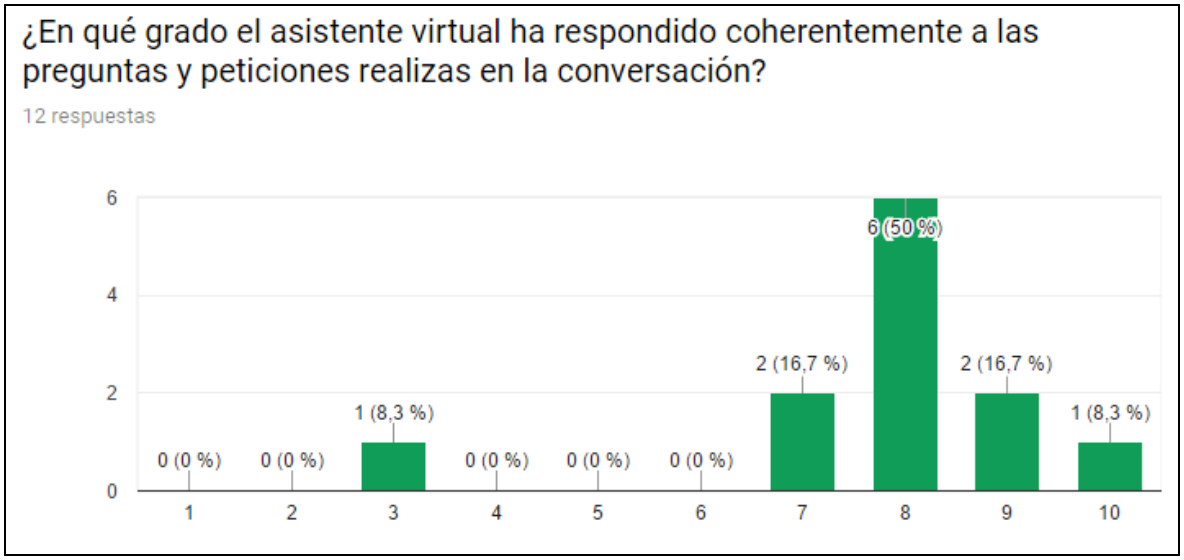
En esta sección se debe rellenar un cuestionario sobre las mejoras o cambios que se podrían realizar sobre el asistente virtual.

1. ¿Qué aspectos de las respuestas del asistente virtual se podrían mejorar?
2. ¿Qué tipo de acciones se podrían incorporar al asistente virtual?
3. ¿Cómo se podría mejorar el sitio web del asistente virtual?
4. En general, ¿el asistente virtual ha cumplido con las expectativas?
5. Aspectos positivos sobre el asistente virtual y el trabajo realizado
6. Aspectos negativos sobre el asistente virtual y el trabajo realizado

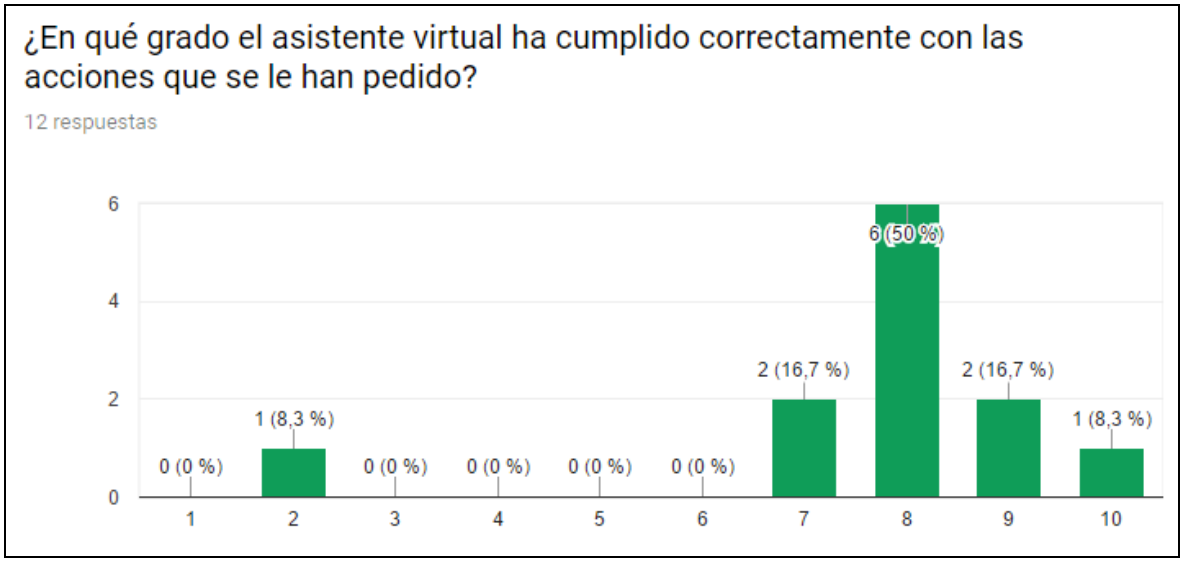
VI. Detalles de la evaluación de Watson AV

Evaluación de Watson AV

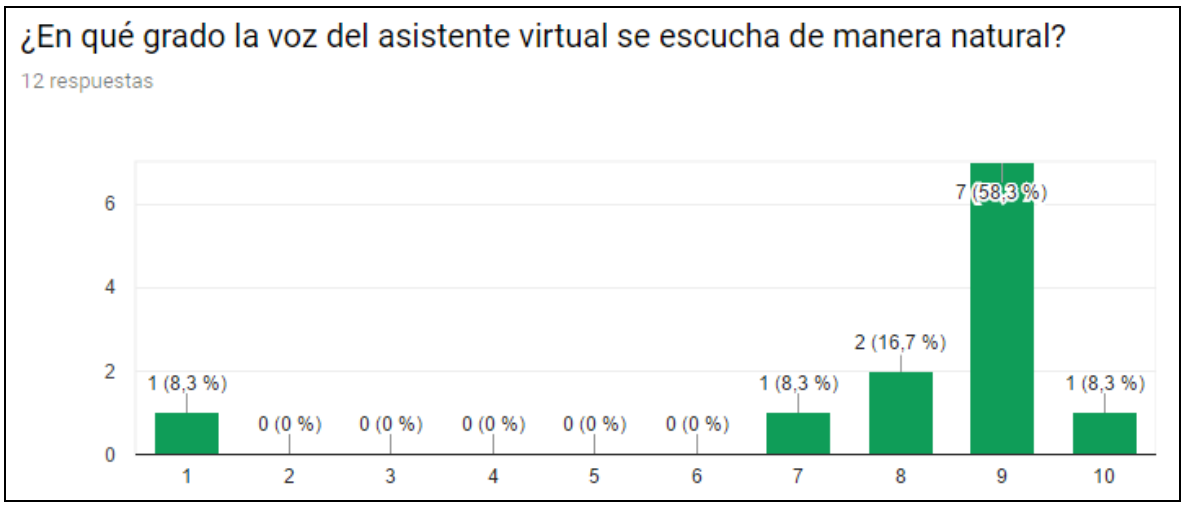
Las siguientes figuras muestran los resultados obtenidos en la sección "Evaluación de Watson AV" del formulario presentado en el Anexo V. Se puede observar que hay una puntuación que se sale de la estadística, esto es debido a un bug que se presentó en el módulo de conversación que impedía que se reconociesen correctamente las intenciones y entidades. Nada más ser notificado fue resuelto.



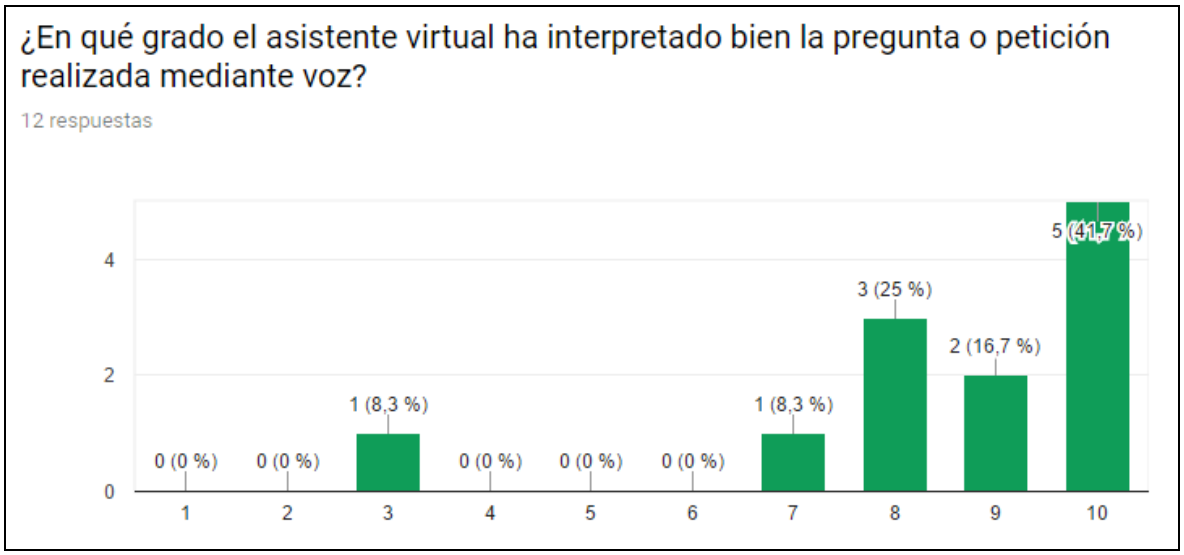
Coherencia en las respuestas



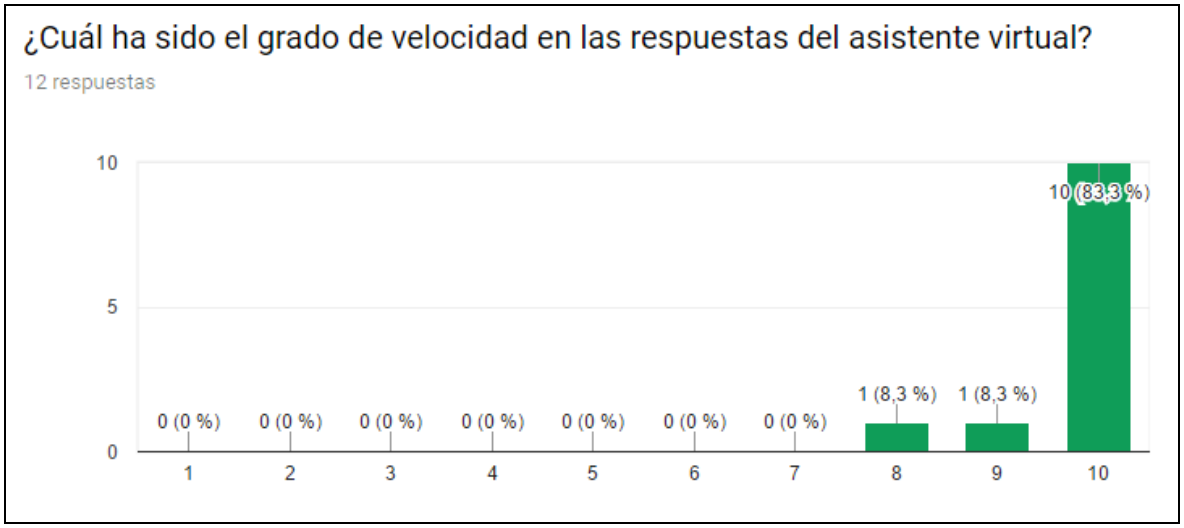
Ejecución correcta de acciones



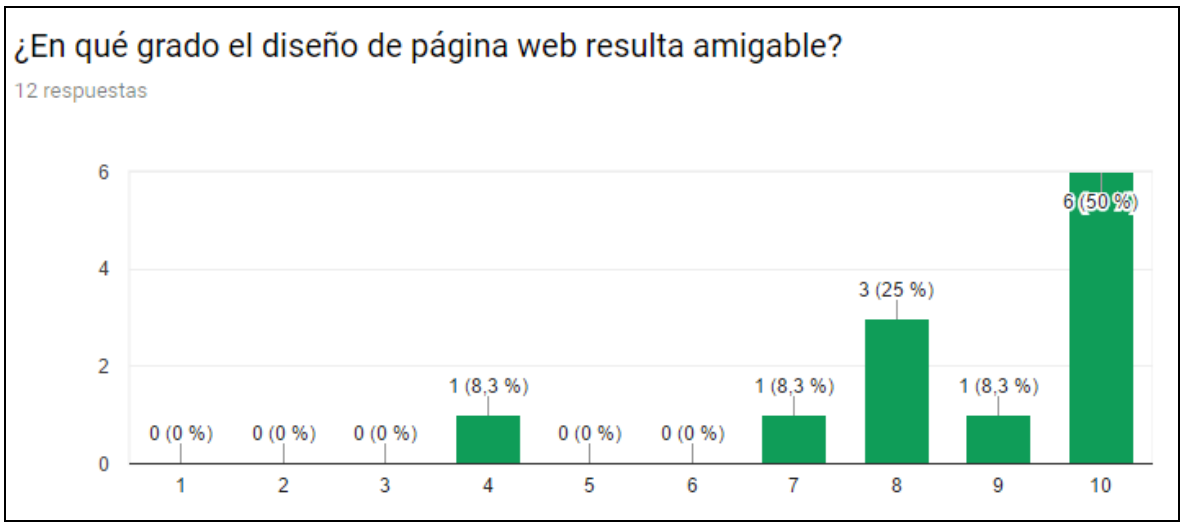
Sonido de voz natural



Interpretación de la pregunta por voz



Velocidad de respuestas



Interfaz gráfica

Mejoras sobre Watson AV

Varias de las respuestas recibidas en esta sección comparten contenido. Lo que se muestra a continuación es una síntesis de las respuestas obtenidas a cada pregunta.

- **¿Qué aspectos de las respuestas del asistente virtual se podrían mejorar?**

  Mejor contextualización y exactitud, además de añadir más variación de respuestas.

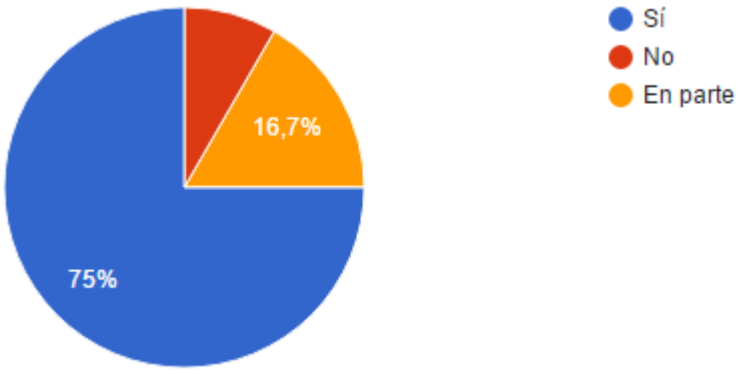
- **¿Qué tipo de acciones se podrían incorporar al asistente virtual?**

  Mayor sabiduría en los temas con los que se trabaja: más acciones relacionadas con la conducción.

- **¿Cómo se podría mejorar el sitio web del asistente virtual?**

  Añadir animaciones que ilustren las acciones del asistente.

- **En general, ¿el asistente virtual ha cumplido con las expectativas?**



El gráfico muestra la satisfacción general de los usuarios, la zona azul corresponde a 9 de los 12 usuarios que realizaron las pruebas y la zona naranja a 2 de ellos. La zona marcada de rojo corresponde a aquella prueba en la que se identificó el bug comentado al inicio de este anexo.